software AG

System Administration

Alfabet Reference Manual

Documentation Version Alfabet 10.11.2

Conventions used in the documentation

Convention	Meaning
Bold	Used for all elements displayed in the Alfabet interface including, for example, menu items, tabs, buttons, dialog boxes, page view names, and commands. Example: Click Finish when setup is completed.
Italics	Used for emphasis, titles of chapters and manuals. this Example: see the *Administration* reference manual.
Initial Capitals	Used for attribute or property values. Example: The object state Active describes...
All Capitals	Keyboard keys Example: CTRL+SHIFT
File > Open	Used for menu actions that are to be performed by the user. Example: To exit an application, select File > Exit
< >	Variable user input Example: Create a new user and enter <User Name>. (Replace < > with variable data.)
(i)	This is a note providing additional information.
(tree icon)	This is a note providing procedural information.
(eye icon)	This is a note providing an example.
(warning icon)	This is a note providing warning information.

Table of Contents

Chapter 1:   Welcome to Alfabet

Purpose of This Document

This document describes in detail the installation and operation of the components of the Alfabet software.

This document supports:

- **Installation**: The installation process is supported with documentation covering various installation scenarios and how to upgrade an existing installation to a new release.

- **Daily operations**: The execution of daily operations is supported with documentation covering required operational processes.

- **Special operations** like system restore, fallback scenarios, and upgrade.

- **All other active and proactive tasks** that may be necessary to fulfill the company's requirements. Such tasks include, for example, interfacing with external databases, sending email notifications, and batch-processing data.

(i) The instructions in this manual are limited to Alfabet-specific operations. This manual assumes basic knowledge about platform technology such as database management, handling of Windows operating systems, Web server management.

⚠ Please note that changes to the Alfabet database structure shall only be performed via the official Alfabet components such as Alfabet Expand or Alfabet Administrator. Software AG cannot be held responsible for any damage done to the Alfabet database by changes made by customers with direct access to the database (for example, via direct execution of SQL commands such as `DROP`, `ADD`, etc. that alter the database structure).

This manual serves the following groups in your enterprise:

- **Application Support Team, System Administrators, and Database Administrators** as an instructional manual to operate Alfabet.

- **Operations Managers** to understand which operations must be executed.

- **Technical Writers** as a basis to create a System and Operations Handbook.

Related Documents

The following PDFs are useful for the same audience:

- *Technical Requirements*: Provides an overview of the most important technical prerequisites for the operation of Alfabet.

- *Configuring Alfabet with Alfabet Expand*: Provides a detailed description of the configuration options for the Alfabet solution that can be defined with the tool Alfabet Expand. This

configuration is normally done by the solution designer, but some configuration tasks described in the reference manual *Configuring Alfabet with Alfabet Expand* are necessary for procedures described in this manual.

- *Alfabet RESTful API*: Provides the information required to build a RESTful client that communicates with the RESTful API of the Alfabet Web Application in order to access to the Alfabet database.

- *ARIS- Alfabet Interoperability*: Provides information about the implementation and operation of a direct interface between Alfabet and Software AG 's business process planning application ARIS.

- *Web Services for Alfabet*: Provides an overview of Alfabet Web Services. Alfabet Web Services provide an SOAP-based access to the Alfabet database for external applications.

- *Alfabet Data Integration Framework*: Provides detailed information about the configuration of data import and export schemes triggering batch data exchange between the Alfabet database and either Microsoft® Excel®, XML or CVS files or external databases.

- *Alfabet- CentraSite Interoperability*: Provides detailed information about the implementation and operation of an interface with Software AG 's CentraSite product in order to synchronize the planning of technical services in Alfabet with the implementation and run-time governance of technical services in CentraSite.

- *Designing Guide Pages for Alfabet*: Provides a description about the generation of customized navigation pages for Alfabet.

Additional documentation of Alfabet functionalities:

- Alfabet Expand Online Help

- Alfabet Online Help

- ADIF Online Help for Alfabet meta-model

- Alfabet Reference Manuals:

  - Alfabet Glossary

  - *Getting Started with Alfabet*

  - *Designing IT Landscape Diagrams in Alfabet*

  - *Configuring Alfabet with Alfabet Expand*: Please note that this reference manual has been updated to include information about the configuration of geo map reports.

  - *Configuring Alfabet with Alfabet Expand- Appendix*

  - *User and Solution Administration*

  - *Configuring Evaluation and Reference Data in Alfabet*

  - *Alfabet Meta-Model*

  - *Enterprise Architecture Management*

  - *Portfolio Management Basic*

  - *Portfolio Management Advanced*

  - *Portfolio Management Complete*

- The following reference manuals are structured according to the current go-to-market capability packages provided by Software AG. New reference manuals providing a methodological approach are being written and will be completed and published in an upcoming release. Until then, the following interim reference manuals are available that list each capability available in the package as well as the object classes assigned to the capability and the views available in the standard object profile of each object class.

    - IT Governance, Risk, and Compliance Reference Manual

    - IT Planning Basic Reference Manual

    - IT Planning Advanced Reference Manual

    - IT Planning Complete Reference Manual

- Examples of Configured Reports in the Showcase Database

Service and Support

Please open a ticket in the Empower eService for any service request as well as all non-standard support incidents such as training requests, scripting, or data integration:

https://empower.softwareag.com

When you submit a ticket for a service request, you should include the main release number and patch version of your Alfabet product. This information can be accessed by clicking **Help** < **About Alfabet**. Tickets will be recorded and transferred to the relevant team.

Empower eService also includes:

- tracking ticket statuses

- local telephone numbers for support.

In addition to the local support telephone numbers, you can use the following toll-free number:

+800 2747 4357

Chapter 2:   System Overview

Alfabet is a Web-based client-server application with an underlying database.

On server side, the Alfabet components consist of the Alfabet database, the Alfabet Web Application and the Alfabet Server:

- Alfabet database

  The Alfabet database is installed on a third-party database server. The database server can be a Microsoft® SQL Server® or an Oracle® database server. The database server is not delivered with the Alfabet components. The database server can be installed either on the same host as the Alfabet Web Application or on a separate host. Database clusters can be used.

- Alfabet Web Application

  The Alfabet Web Application provides access to the Alfabet database and functionality to the clients.

  The Alfabet Web Application is a Microsoft.NET Framework 4.8 application and requires a Microsoft Windows operating system and must be installed on a Microsoft® Internet Information Services® Web Server.

  Multiple Alfabet Web Application s can access the same Alfabet database in parallel. Load balancing is supported to make the system highly scalable. Implementation of multiple Alfabet Web Application s also supports implementation of multiple different access modes for the clients.

- Alfabet Server

  The Alfabet Server is required to execute asynchronous processes triggered by user activity on the Alfabet user interface. The Alfabet Server is a Microsoft.NET Framework 4.8 application and requires a Microsoft Windows operating system. The Alfabet Server can be installed either on the same host as the Alfabet Web Application or on a separate host.

If the server-side components are installed on different host, all hosts must have the same server time.

On client side, users can access Alfabet via a web browser. Access to Alfabet is highly flexible concerning the use of browsers and operating systems and can be performed both from desk top computers and mobile devices.
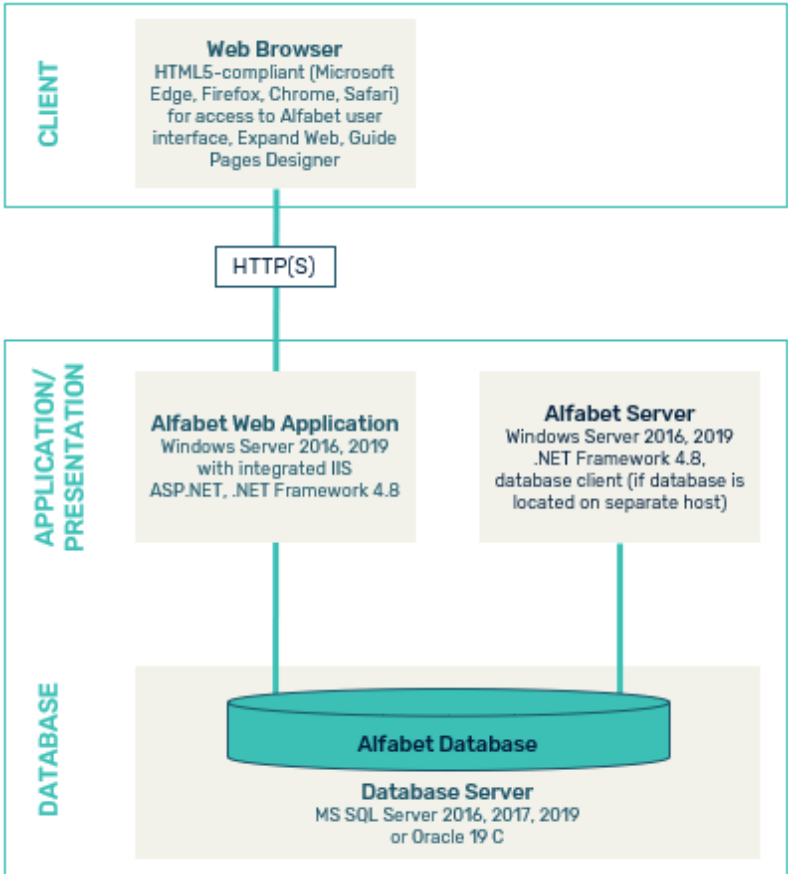
FIGURE: Alfabet components on server and client side for working with Alfabet

Tools and Functionalities for System Administration and Configuration

Alfabet is a highly complex and customizable application. The components that are required for working with Alfabet are therefore extended by a number of tools for configuration, administration, and maintenance of the main Alfabet application.

The tools listed in the following table are optional components. Their availability is regulated by your enterprise's licensing agreement with Software AG.

Use the Configuration Tool...	In Order To...
Alfabet Administrator	The Alfabet Administrator is the main tool for system administrators. It is used for performing administrative tasks concerning the Alfabet components and managing the Alfabet database. The functionalities provided by the Alfabet Administrator are described in this manual.
Alfabet Expand	Alfabet Expand is the main tool for system designers. It provides the main functionalities for the configuration of the Alfabet solution. Many Alfabet functionalities as well as administrative tasks are based on a specific solution configuration defined in Alfabet Expand. Alfabet Expand is available both as an application and as web-based interface that is integrated into the Alfabet Web Application. The web-based interface of Alfabet Expand is only available for Cloud customers and not included in the installation described in this reference manual. For detailed information about the configuration tool Alfabet Expand, see the reference manual *Configuring Alfabet with Alfabet Expand*.
Guide Pages Designer	The Guide Pages Designer allows start pages to be configured that provide users with access to relevant functionalities in the Alfabet user interface. Different navigation pages can be configured for different user groups. The Guide Pages Designer is accessible via Alfabet Expand. Please note that access to the Guide Pages Designer requires that a license key for the Guide Pages Designer is added to the server alias configuration used to access the Alfabet database. For information about the Guide Pages Designer, see the reference manual *Designing Guide Pages for Alfabet*.
Batch processing tools	A number of administrative tasks require the execution of command line tools. In addition, some functionalities that can be triggered or scheduled via the Alfabet user interface can alternatively be executed by command line tools. Command line tools are available for the following tasks: • Administrative tasks like installing a server service.

Use the Con-figuration Tool...	In Order To...
	• Batch jobs that must be executed in regular intervals to enable functionalities of the Alfabet application such as sending reminder emails, re-calculating indicators, or updating the full-text search index. • Batch data manipulation of the data stored in the Alfabet database. • Triggering data exchange with external applications (for example, via the Alfabet Data Integration Framework (ADIF).) Executables for batch jobs can be started in a command line or by means of a Windows® batch job. The documentation of the execution of batch processing tools is part of this manual and provided in general in the section About Batch Utilities for Alfabet and in the context of the documentation of the respective functionality triggered by the batch processing tools.
ARIS- Alfabet Interoperability Interface	The ARIS- Alfabet Interoperability Interface triggers the data exchange between the Alfabet database and the database of Software AG 's business process modelling application ARIS. Technically, the ARIS- Alfabet Interoperability Interface is implemented in two ways that can be used alternatively: • Data exchange can be performed via Web Services, which are integrated into the Alfabet Web Services on the Alfabet side, and a batch processing tool that requires a connection to the Alfabet database via an Alfabet Server. • Data exchange can be performed via a RESTful API implemented on both applications. For information on the ARIS- Alfabet Interoperability Interface, see the reference manual *ARIS- Alfabet Interoperability*.
Alfabet Web Services	The Alfabet Web Services provide an interface for data exchange between the Alfabet database and third-party applications. For information on the Alfabet Web Services, see the reference manual *Web Services for Alfabet*.

Some of the tasks that are relevant for system administrators require configurations that are carried out in the Alfabet user interface. If this is the case, the required configuration steps are described in conjunction with the documentation of the respective task. For general information about how to access the Alfabet user interface and work with Alfabet, see the reference manual *Getting Started with Alfabet*.

The Alfabet Administrator and the configuration tools included in the customer license are usually located in the same installation directory as the main Alfabet installation and are automatically installed during the installation procedure. The image below displays the complete installation of configuration tools results for the Alfabet application:
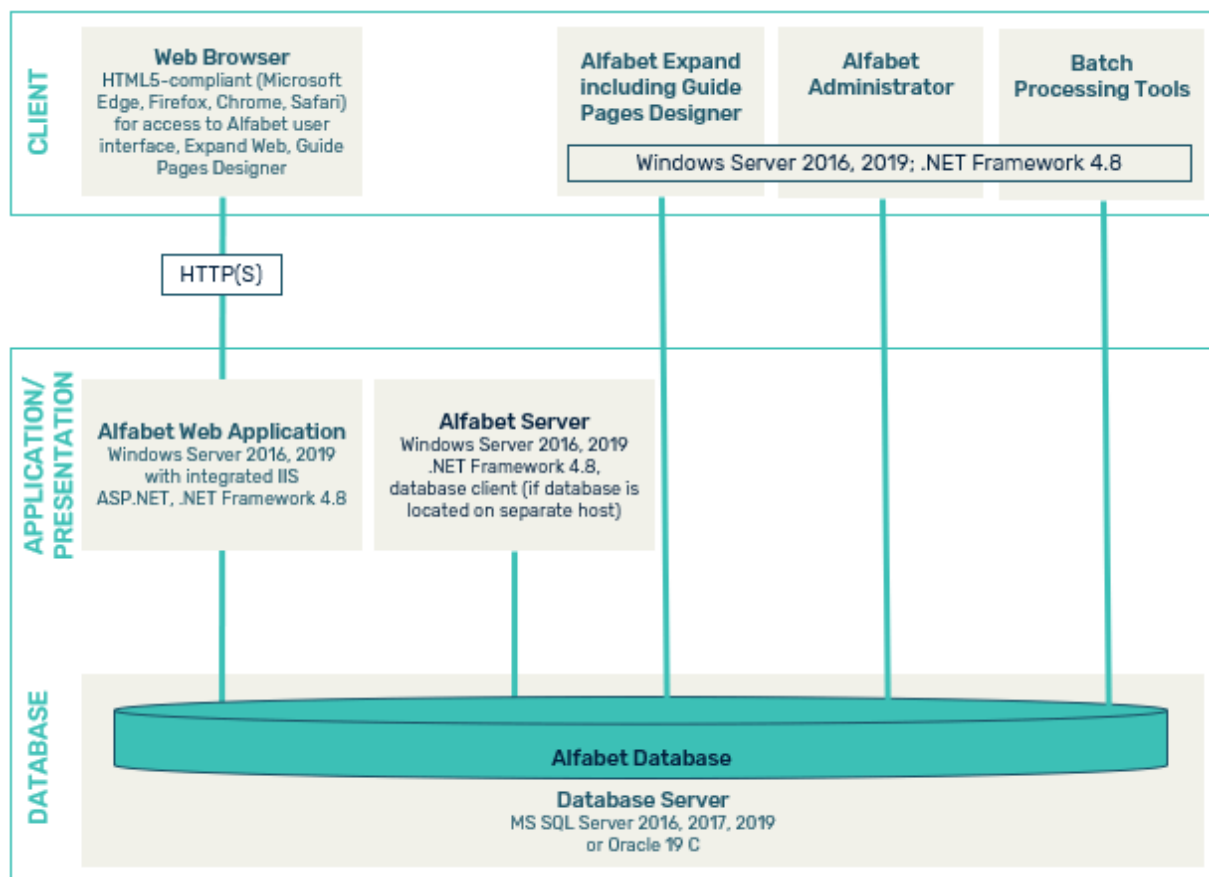
FIGURE: Setup of the Alfabet components including configuration tools

The Alfabet Administrator and the configuration tools directly access the Alfabet database. Therefore, it is recommended that access to the configuration tools is limited toa small number of users that are responsible for maintaining and configuring the solution. Details about the configuration required to access the configuration tools is provided in the section Enabling a User to Access Tools for Configuration and Administration.

Best Practice Installation and Workflow

The best practice installation proposed in the following shall ensure a reliable and stable production environment. It is a basic scenario that does not take special demands for the production environment into account such as running multiple Alfabet Web Application s in parallel for a multi-lingual user community.

In the following section, the production environment itself and the surrounding best practice environments consist each of one simple Alfabet instance (one Alfabet database with one Alfabet Web Application).

It is recommended that you install at least three Alfabet instances in parallel to the production environment for the following purposes:

- **Failover**

  Alfabet is not a critical system. Nevertheless, company standards might require that a Failover scenario for Alfabet is available.

- **Configuration and Testing**

  The Alfabet application is highly configurable. Configuration of new features or required changes to the existing configuration is primarily done with the tool Alfabet Expand. The Alfabet Expand application must connect directly to the Alfabet database to provide full functionality including the Guide Pages Designer. Alfabet Expand includes a mechanism to open the Alfabet user interface in a view mode or configuration mode that is independent of the Alfabet Web Application. Optionally, the Alfabet Web Application may be required in the configuration environment if the Web-based version of Alfabet Expand is used for configuration of the customer's solution.

  To ensure a stable and performant production environment, it is recommended that configuration and testing is carried out in separate environments. This requires the setup of separate instances for configuration, testing and production environment. Configuration and testing is done each on a copy of the production database. Software AG provides mechanisms to export custom configurations from a database and apply the configuration to an existing database without overwriting the existing object data.

  ⚠ A single Alfabet production instance is defined by one Alfabet database. It may be complemented by up to 4 application servers and 4 web servers connecting to this single Alfabet database. Test and development environments require purchase of the intended number of non-production instance licenses and are not covered by a production instance license. Setup of a best practice environment for configuration and testing requires the appropriate number of non-production licenses.

The following figure gives an overview over the complete installation in a best practice scenario. On server side, the different environments mentioned above are displayed. On the client-side, access to the different environments depends on the role of the user accessing the Alfabet components on the server-side.
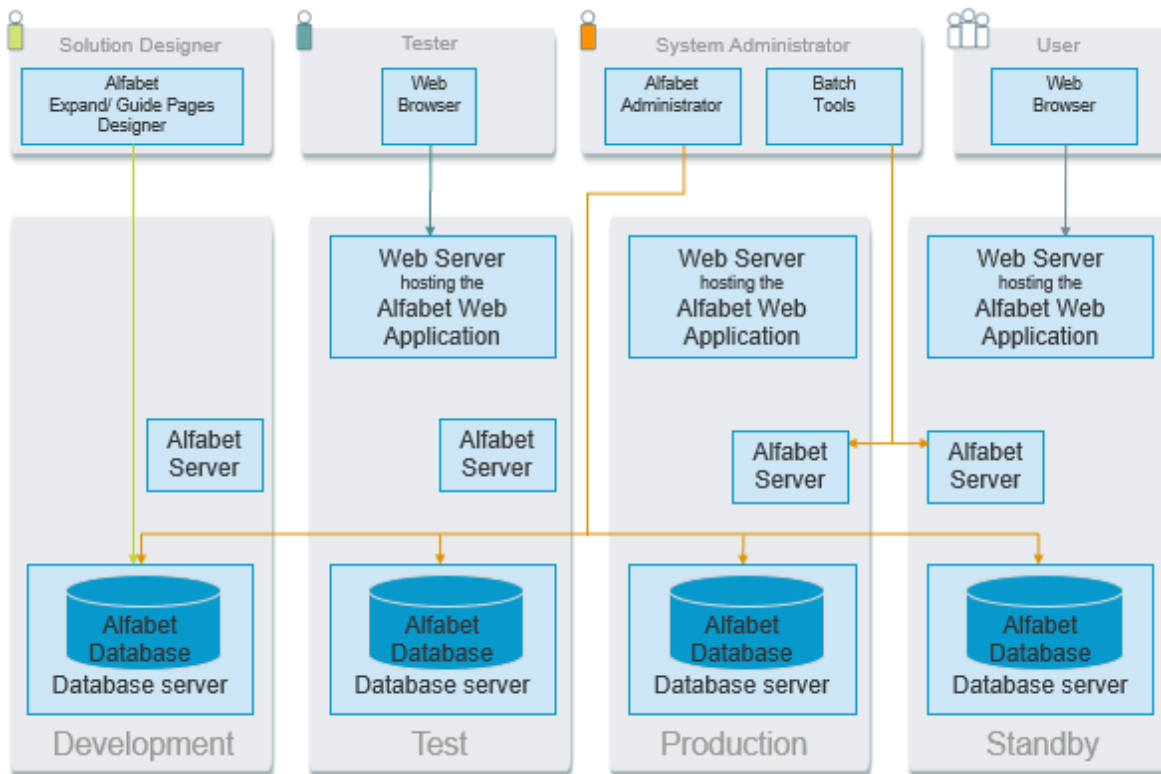
FIGURE: Overview of best practice installations and the access of roles on the client-side to the Alfabet components on the server-side

The following roles are involved in the use and maintenance of the Alfabet application:

	Role	Task	Tools
	Solution Designer	The solution designer configures the Alfabet solution according to the company's demands. This includes configuration of user profiles and access permissions and activation of functionalities as well as the visibility of functionality in the user interface. The solution designer works in the Configuration environment.	Alfabet Expand, Guide Pages Designer
	Application Administrator	The application administrator is responsible for the installation and maintenance of the Alfabet solution. He/she performs migrations, Alfabet-related archive and restore actions on the database and sets up batch jobs. The application administrator is responsible for all installations in the proposed best practice scenario. He/she is, for example, responsible for merging new configurations to the existing configuration in the test and production environment.	Alfabet Administrator, Alfabet Server with batch processing tools, Alfabet Expand, Guide Pages Designer
	Database Administrator	The database administrator is typically not only responsible for the Alfabet database, but also for the administration of the company's database servers and all databases on the database servers. For the Alfabet database, the database administrator is involved in database setup and backup independent of the Alfabet environments.	none
	Web Administrator	The Web administrator is responsible for the company's web servers. The Web administrator is involved in the configuration of the web server to host the Alfabet Web Application or the Alfabet Web Services.	none
	Tester	The tester is responsible for quality assurance. All configuration and newly acquired interfaces or releases need to be tested before being applied to the production environment. Tests are performed in a test environment simulating the conditions of the production environment.	Access via the Alfabet Web Application using a Browser
	User	Users are responsible for a variety of data entry, maintenance and control functionalities performed via the Alfabet user interface. They work in the production environment.	Access via the Alfabet Web Application using a Browser

Depending on the structure of your IT department, there may be other people involved such as the administrator of the company's network.

Chapter 3:   Installation

This chapter describes the entire installation process for a typical installation scenario which includes in-stallation of one Alfabet database and one Alfabet Web Application and the installation of the tools for ad-ministration and configuration of the application.

The following information is available:

Scope of Delivery

Software AG delivers the following for the initial installation:

- • Alfabet software components

- • Serial numbers for Alfabet software components

- • An initial database for testing purposes as database archive file in Alfabet specific format.

- • An AMM file for update of the meta-model that is required for installation of upgrades.

- • Documentation (PDFs) for Alfabet

The CD contains the software required for the Alfabet components. Serial numbers are either stored on the CD or delivered in the text of the delivery email. Each serial number carries out a different type of installation. In this way, the same installation CD can be used to install different components.

Depending on the serial numbers delivered, the following components may be available:

Type	Description
Alfabet Web Application	The Alfabet Web Application necessary to access Alfabet from the client host.
Alfabet Administrator	Tool for the administration of the Alfabet Server and the Alfabet database.
Alfabet Expand	Tool for the configuration of the Alfabet solution according to the customer needs. Alfabet Expand is available as an application and as a Web-based tool integrated into the Alfabet Web Application.
Guide Pages Designer	Web-based tool to create and edit customized navigation pages for the Alfabet interface. The Guide Pages Designer is an integral part of the Web-based version of Alfabet Expand and the Alfabet Expand application.
Alfabet Server	The Alfabet Server executes asynchronous processes either triggered by a user on the Alfabet user interface or triggered by batch jobs that are delivered by Software AG For example, asynchronous processes are required to send emails, start workflows, or re-calculate indicators defined to be automatically calculated.
Alfabet Web Service	The Alfabet Web Service allows direct access to the data stored in the Alfabet database from external programs. The Alfabet Web Service to read data from the Alfabet database is included in the license for custom report generation. The Alfabet Web Services to write data to the database is included in the license for the Alfabet Data Integration Framework (ADIF).
Alfabet Data Integration Framework	The Alfabet Data Integration Framework (referred to as ADIF) is a technology to batch import, export, and manipulate very large amounts of data in the Alfabet database. Its interface is highly configurable by means of native SQL commands in combination with conversion procedures delivered by Software AG that ensure database consistency. Export can be performed from external databases, Microsoft® Excel® files, XML files, or CSV files.
ARIS- Alfabet Interoperability Interface	The ARIS- Alfabet Interoperability Interface synchronizes data in the databases Alfabet and Software AG 's business process modelling application, ARIS. In addition, links are set between the user interfaces of both applications to ease parallel work in both applications.
Alfabet REST API	The Alfabet RESTful API provides easy access to the content in the Alfabet database. The API is designed as a web service architecture based on the Representational State Transfer (REST) software architecture type. It can be used to get information about the structure of the object class model or information about objects stored in the Alfabet database. It also provides an endpoint to update data in the Alfabet database.

Overview of the Installation Process

The following must be performed in the sequence described below in order to successfully install the Alfabet components:

Role	Task
System Administrator	Provide a server host infrastructure matching the technical requirements for Alfabet as described in the reference manual *Technical Requirements*. If antivirus software is used, the requirements for Directories Used by the Alfabet Components Impacted by Antivirus Software must also be considered.
Web Administrator	Configure the required Web Server Roles on the Web server host. For more information, see Required Web Server Roles.
Database Administrator	Set up a database for Alfabet and configure access permissions. For more information, see Pre-Installation Database Management Tasks.
Application Administrator	Install the Alfabet components. For more information, see Basic Installation of the Alfabet Components.
Application Administrator	Configure the Alfabet Web Application. For more information, see Basic Configuration of the Alfabet Components.
Application Administrator	Apply Alfabet structure to the Alfabet database. For more information, see Setting up the Initial Alfabet Database.
Application Administrator/ Web Administrator	Install the Alfabet Web Application on the Web server. For more information, see Setting Up the Alfabet Web Application .
Application Administrator	Test the installation. For more information see Testing the Installation.
Application Administrator	Configure the Alfabet components for Administration and Configuration. For more information, see Installation of Tools for Configuration and Administration in Alfabet .

Required Web Server Roles

Web Server Roles must be added to the Web server hosting the Alfabet Web Application.

1) On the Web Server host, click the **Start** [⊞] icon that appears when you move the mouse to the lower left corner and click on the **Server Manager** [icon] icon to open the Server Manager.

2) In the menu on the upper right of the Server Manager, select **Manage** > **Add Roles and Features**.

3) In the **Add Roles and Features Wizard**, select the installation type and the Web server host in the first three pages of the wizard and proceed to the **Server Roles** page, using the button **Next** of the wizard.

4) In the **Server Roles** page, expand **Web Server (IIS)**:
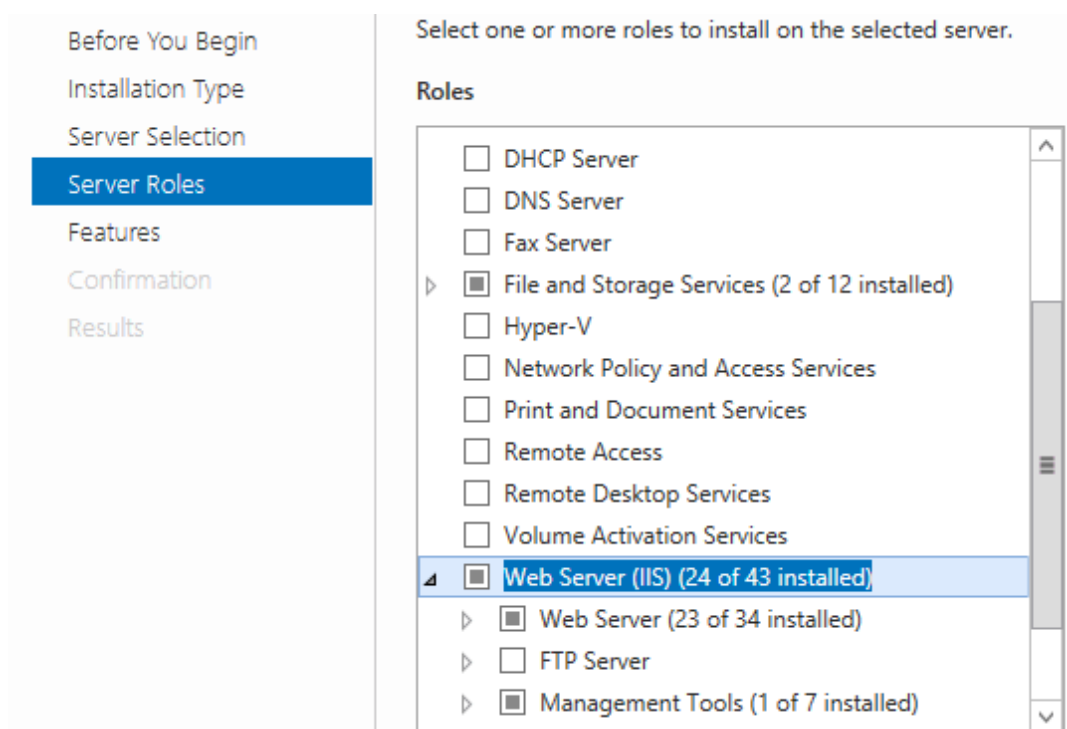
5) Activate the check boxes of at least the following roles:

```
[X] Web Server installed

    [X] Common HTTP Features

        [X] Default Document

        [X] HTTP Errors

        [X] Static Content

        [X] HTTP Redirection

    [X] Health and Diagnostics

        [X] HTTP Logging

        [X] ODBC Logging

    [X] Performance
```

```
                    [X] Static Content Compression

                    [X] Dynamic Content Compression

            [X] Security

                    [X] Request Filtering

            [X] Application Development

                    [X].NET Extensibility 4.7

                    [X] ASP.NET 4.7

                    [X] ISAPI Extensions

                    [X] ISAPI Filters

        [X] Management Tools

                    [X] IIS Management Console
```

6) In the section **Web Server installed** > **Security**, activate the check box **Windows Authentication (SSO)** to use Windows sign-on or **Basic Authentication** to use standard login.

> A detailed description of the available authentication modes and the implementation thereof is given in the section Configuring User Authentication.

7) Click the button **Next** to proceed to the **Features** page.

8) Activate the check boxes of at least the following features:

```
        [X] Web Server installed

            [X].NET Framework 4.5 Features

                [X].NET Framework 4.5

                [X] ASP.NET 4.5
```

9) Click **Next**.

10) Click **Install**. The roles and features are installed.

11) Click **Close** to close the wizard.

After having installed the required roles and features, the Web Server should be restarted. You can restart the Microsoft® Internet Information Services® via the Internet Information Services (IIS) Manager. You can open the IIS Manager in the menu of the Server Manager by clicking **Tools** > **Internet Information Services (IIS) Manager**. In the explorer of the IIS Manager, select the Web Server and click **Restart** in the **Manage Server** pane on the right.

Pre-Installation Database Management Tasks

Prior to installing the Alfabet components, a database must be created on the third-party database server and login must be configured for the database.

The Alfabet components must log in to the database as database owner.

> If it is not possible to grant database owner rights to the Alfabet components during normal operations, the following minimum of rights must be granted to the Alfabet components:

For Oracle:

- SELECT ON "PUBLIC".V$LOCKED_OBJECT

- SELECT ON "PUBLIC".USER_OBJECTS

- SELECT ON "PUBLIC".V$SESSION

- ALTER SYSTEM

- CREATE TABLE

- CREATE SESSION

- CREATE TRIGGER

- CREATE PROCEDURE

- UNLIMITED TABLESPACE

For Microsoft SQL Server:

- GRANT DELETE

- GRANT EXECUTE

- GRANT INSERT

- GRANT SELECT

- GRANT UPDATE

- GRANT ALTER ANY SCHEMA

- GRANT CREATE FUNCTION

- GRANT CREATE PROCEDURE

- GRANT CREATE TABLE

- GRANT CREATE VIEW

The following documentation does not repeat the rights concept described above each time database access is mentioned. Instead, the documentation is limited to the recommended database owner concept.

⚠ Please note however that database owner rights are mandatory to perform meta-model updates and to create or restore databases. During these processes, the database is run in a restricted mode that does not allow any other processes or users to access the database. If database owner rights shall not be granted during normal operations for security reasons, the database permissions must be changed prior to and after migration.

The database creation depends on the server that the database is installed on and is typically performed by a database server administrator familiar with the workflow for database creation and administration. The following information must be delivered to the application administrator installing the Alfabet components to configure the connection to the Alfabet database:

- Database server name

- Database name

- User (Schema for Oracle Databases; optional, if Windows sign-on is used)

- Password (optional, if Windows sign-on is used)

- The Alfabet-specific database preconditions listed in the following

	Microsoft® SQL Server	Oracle®
Unit to host data of a single Alfabet instance	Database	Schema
Character Set (Recommendation)	Latin1_General (recommended for Western Europe and Northern America)	Unicode (AL32UTF8)
Case-Sensitivity	It is recommended that you use a case-insensitive database. The case-sensitivity settings of the database will affect the behavior of search functionalities and filter settings in reports.	
Rights for the database user used for the Alfabet instance	db_owner role membership Alfabet default schema: dbo	DBA role membership

ⓘ For example, if multiple Alfabet instances are used for a development and a production environment in parallel, the following requirements apply to the setup of the databases:

- All Alfabet instances must have the same character set and case sensitivity settings to avoid inconsistent behavior.

- If Oracle database servers are used, each Alfabet database must be installed on a separate Oracle instance with a separate service.

An overview of the access permission concepts for the connection of the Alfabet components to the database server is provided in the following section. The following information is available:

- Configuring Authentication Between the Alfabet Components and the Database Server

  - Standard Login

  - Windows Integrated Login

Configuring Authentication Between the Alfabet Components and the Database Server

There are two authentication methods for access to database servers. The availability of authentication mechanisms depends on the database server used:

Database Server	Available Authentication Mechanisms
Microsoft SQL Server	Standard Login Windows Integrated Login
Oracle Database Server	Standard Login

Standard Login

Standard login requires a user name and password that is configured on the database server for access to the Alfabet database. The user name and password are then added to the server alias configuration for login to the Alfabet database.

> ⓘ The database login parameters are only stored in the server alias configuration and not added to remote alias configurations. The login information is therefore not available on any client hosts given that no Server alias configurations are added to the AlfabetMS.xml configuration file of the clients.

Standard login is set up as follows. The information describes the tasks that may fall under the responsibility of different personnel.

Role	Task
Database Administrator	Configure the login parameters for the Alfabet database on the database server and inform the application administrator about the following: • The user name and password required for login • The name of the database. For an Oracle® database, use the Net Service Name of the database configured with the Oracle® Net Assistant. For a Microsoft® SQL Server® database, use the notation servername\databasename for a default instance or servername\instancename\databasename for a named instance.
Application Administrator	Configure the Alfabet components to use the login parameter when connecting to the Alfabet database. Configuration is done with the tool Alfabet Administrator. Configuration of the database settings for the Alfabet Server is described in the section Creating a Server Alias for the Alfabet Web Application.

Windows Integrated Login

The Windows-integrated login mechanism of Microsoft® SQL Servers® allows the Windows login credentials defined on the domain controller in the Active Directory (Active Directory Domain Services on the Windows Server) to be used. The login parameters for Windows-integrated login are secured in the Windows registry.

Windows integrated login is set up as follows. The information describes the tasks that may fall under the responsibility of different personnel.

Role	Task
Network Administrator	Configure a service account for the Alfabet components in the Active Directory on the domain controller. The account must have a non-expiring password.
Database Administrator	Create a SQL Server login for the service account. Map the login to a database user in the required database with a role granting the required access permissions.
Application Administrator/ Web Administrator	Set the Identity of the application pool used for the Alfabet Web Application to the service account that has the permissions to access the Alfabet database.
Application Administrator	Make sure that all Alfabet components that log in to the Alfabet database use the service account.

Basic Installation of the Alfabet Components

⚠ Please note that you must have administrator rights in order to install Alfabet. The Alfabet components must be installed in an empty directory. If you have already installed a prior release of Alfabet and want to install the components in the same location, the existing files must be removed from the installation directory prior to the installation of Alfabet 10.11.

The following workflow describes the initial installation of Alfabet. If you want to re-install the Alfabet components in order, for example, to upgrade to a new patch release, the installation must meet requirements that are not described here. In this case, see the section for a description of the installation required for upgrades.

1) Insert the Alfabet CD and start the installation via auto run or launch the `setup.exe` file.

2) The **Install Shield Wizard** opens.

3) Click **Next** to continue.

4) Select the checkbox of the option **I accept the terms of the license agreement** and click **Next** to continue.

5) Enter a user name, company name and the serial number delivered by Software AG. The serial number determines which features are available for installation.

> ℹ For information about serial numbers, see the section Scope of Delivery.
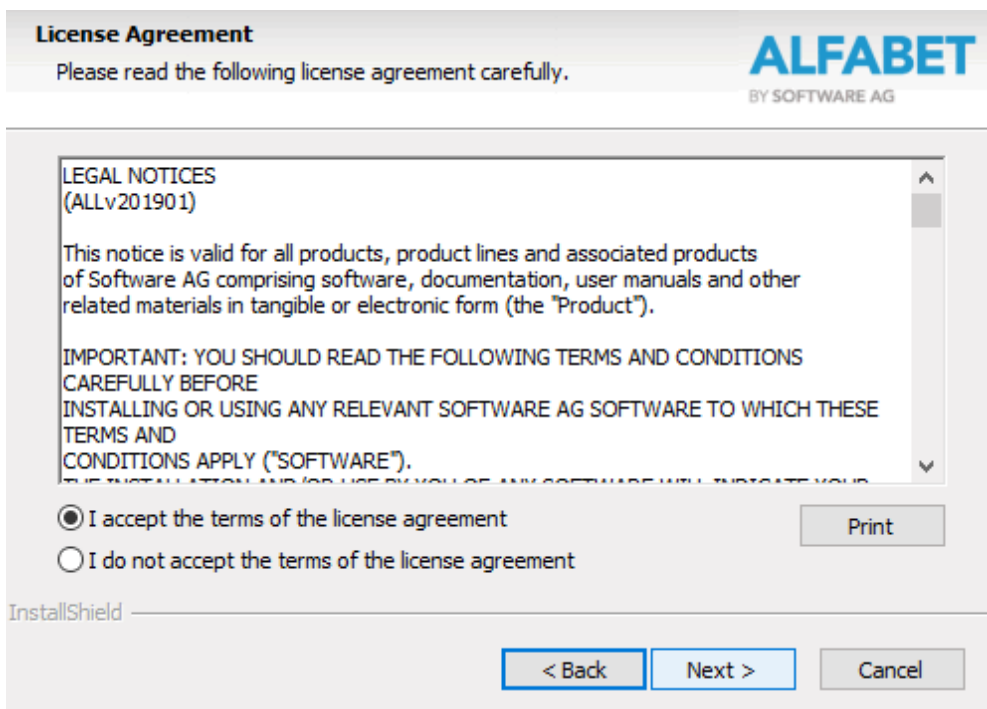
6) Click **Next** to continue.

7) Select **Custom** and click **Next** to continue.

- **Standard** installs all available features. This is recommended for the installation of all Alfabet components included in your license.

- **Custom** allows you to choose features to install. It is recommended that you select this option even if all components will be installed. This allows you to review whether the installation includes all components that are required.

8) The default installation folder is displayed. If you want to install the Alfabet components in another location, click the **Change** button and select a destination folder for installation. If the destination folder does not exist, it will be created.

9) Click **Next** to continue.
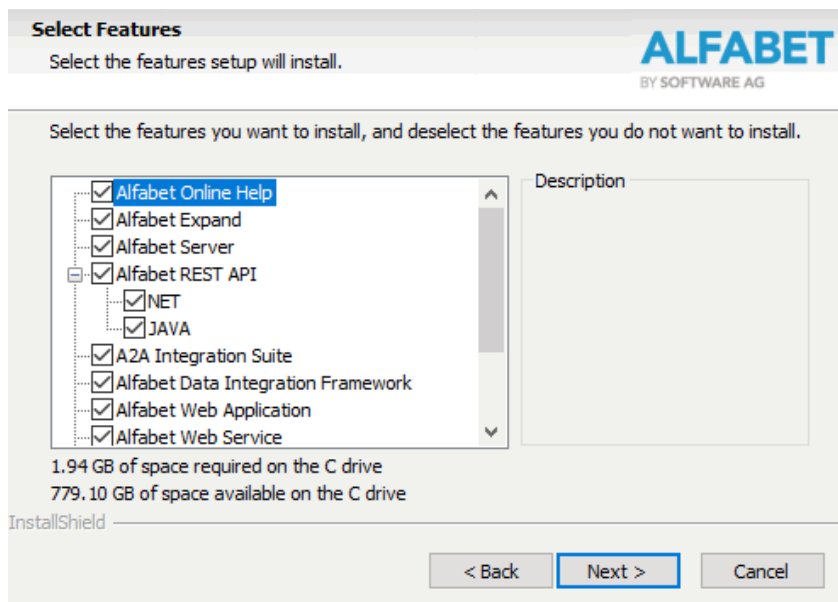
10) Control whether all required components are checked. Select or deselect components, as needed.

(i) The Alfabet Online Help is an integral part of the Alfabet Web Application and cannot be deselected if the Alfabet Web Application is selected for installation. However, you can install the Alfabet Online Help separately with access via a Web server without installing the Alfabet Web Application.
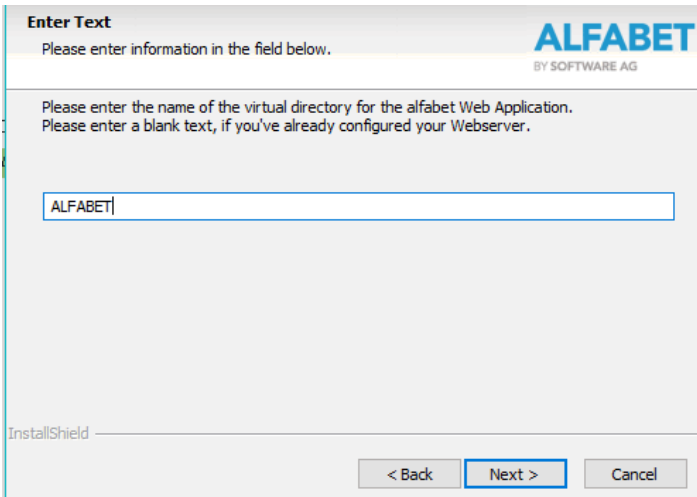
11) Click **Next** to continue.

12) Select a folder to install the Alfabet Web Application files.

(i) Typically, the sub-folder `AlfabetWebApplication` located in the root installation directory is used. If the folder is not available, it will be created automatically during installation. The Alfabet Online Help files are located in the `Help` subdirectory of the Alfabet Web Application directory.

13) Click **Next** to continue.

14) You can create an application directory for the Alfabet Web Application and/or Alfabet Online Help in the Internet Information Services® subordinate to the default Web site. To do so, carry out one of the following steps:

- To create the application directory with the setup procedure: Enter a directory name in the field of the installation shield. A directory with the specified name will be created in the Internet Information Services® during installation of the Alfabet components. The default value is Alfabet. Click **Next** to continue.

- If you have already created the application directory manually: Leave the field blank. Click **Next** to continue.

> ⓘ After installation, you must configure the server alias of the Alfabet Web Application to use the application directory. For information about configuring the access to the application directory, see the section Creating a Server Alias for the Alfabet Web Application.

15) Click **Next** to continue.

16) Select a directory to install the index files required for the full-text search functionality available for Alfabet. The search index for the Alfabet online Help is created during installation in the `HelpSearchIndex` subdirectory of the selected index directory. Additional search indexes generated while working with Alfabet will be located in other sub-directories in the selected directory. After installation, you must configure the Alfabet Server to access this folder to find the search index files.

> ⓘ For information about configuring the access to the search index files, see the section Creating a Server Alias for the Alfabet Web Application.

17) Click **Next** to continue.
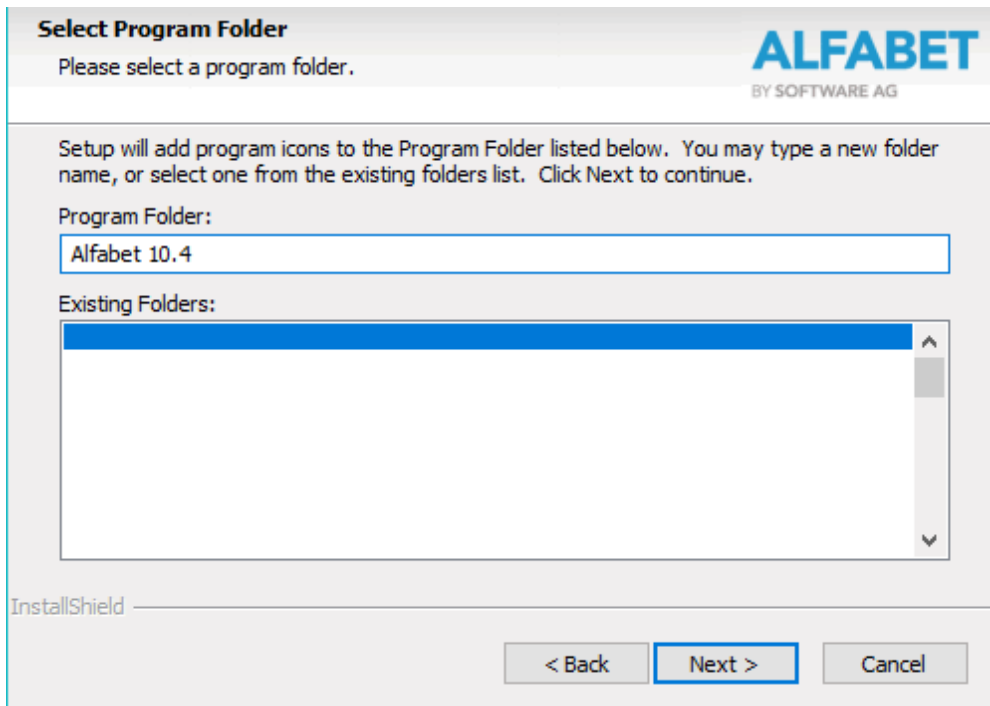
18) Enter the name of the folder where Alfabet will be located. This will be displayed in the Windows®
This will be displayed in the Windows® **Apps** window as section heading.

19) Click **Next** to continue.

20) Click **Install** to start the installation.

21) The progress of the installation procedure is displayed.

(i)   The installation of the Alfabet Online Help may take some time.

22) After installation, additional required files are created and folder permissions are set.

After installation, the following window of the installation shield wizard will be displayed:

23) Click **Finish** to exit the wizard.

Basic Configuration of the Alfabet Components

The following information is available:

- [Understanding the Configuration of the Alfabet Components](#)

- [Creating a Server Alias for the Alfabet Web Application](#)

- [Creating a Server Alias for the Alfabet Server](#)

- [Creating a Remote Alias](#)

- [Configuring the Alfabet Web Application to Hand Over Processes to the Alfabet Server for Execution](#)

Understanding the Configuration of the Alfabet Components

The Alfabet platform components are configured in the `AlfabetMS.xml` configuration file. An `AlfabetMS.xml` configuration file must be available in the `Programs` subdirectory of the Alfabet components.

In addition to the `AlfabetMS.xml` file, the main configuration file `web.config` as well as the subordinate configuration files `Alfabet.config` and `AppSettings.config` are required for the Alfabet Web Application. Example files are provided with the installation. For more information, see the section [Configuring the web.config Files for the Alfabet Web Application](#).

An `AlfabetMS.xml` file can contain several different configurations for the Alfabet components. Each configuration is identified by an alias. There are two different types of aliases:

- Server Alias for the configuration of the Alfabet components that directly connect to the Alfabet database. If different components connect to the same database in parallel, each component must connect with a different server alias. Each server alias must have a different name. All other settings may be identical. A different server alias per component is required in order to enable communication between the components.

- Remote Alias for the configuration of components that connect to the Alfabet Server via .NET remoting.

    ⚠ The concept of direct connections to the Alfabet Server via a remote alias configuration will be deprecated in the near future. For new Alfabet installations remote alias configurations should not be used any longer.

The **Is Remote** attribute determines the type of alias configuration.

Alias	Controlled Components	Configuration Includes	Is Remote Attribute
Server Alias	Alfabet Web Application Alfabet Server	For example:	`False`

Alias	Controlled Components	Configuration Includes	Is Remote Attribute
	Alfabet Expand Alfabet batch processing applications	• Specification of connection to the database • TCP port to listen for client requests	
Remote Alias	Alfabet batch processing applications Alfabet Web Application to connect to the Alfabet Server	For example: • Specification of connection to the Alfabet Server	`True`

In the basic configuration of Alfabet components, you need to create the following aliases:

- A Server Alias for the Alfabet Web Application.

- A Server Alias for the Alfabet Server.

- If the Alfabet Web Application is configured to connect to the Alfabet Server via a remote alias, a Remote Alias for the connection of the Alfabet Web Application to the Alfabet Server is required. Please note that this method will be deprecated in the near future and should not be used for new installations.

The server-side `AlfabetMS.xml` file is usually created and modified with the tool Alfabet Administrator. The Alfabet Administrator will be installed automatically together with the Alfabet components. The `AlfaAdministrator.exe` is located in the Alfabet Programs directory and can be accessed in the Windows® **Start** menu.

The Alfabet Administrator automatically reads and writes the `AlfabetMS.xml` configuration file that is located in the same directory as the Alfabet Administrator. All or part of the alias configuration defined in the `AlfabetMS.xml` file configured with the Alfabet Administrator can be copied to additional `AlfabetMS.xml` files for storage in other locations. This is useful if the Alfabet Web Application and Alfabet Expand or the Alfabet Server are located on different hosts and each component shall read an `AlfabetMS.xml` file containing the server alias of the respective tool only. The mechanism for copying part of the alias configuration to separate `AlfabetMS.xml` files is described in the section of the chapter Working with the Alfabet Administrator .

Each instance of any Alfabet component connecting to the Alfabet database must use a different server alias configuration with a unique name and server name setting. All other settings may be identical. In the Alfabet Administrator you can create a server alias as copy from another server alias and change the name and server name setting to create a server alias with the same connection and behavior for each individual tool.

If the Alfabet Web Application and the Alfabet Server are located on different hosts, the alias configurations centrally defined in the `AlfabetMS.xml` accessed via the Alfabet Administrator can then be written in individual `AlfabetMS.xml` files for each application.

Please note that for installations based on a connection of the Alfabet Web Application to the Alfabet Server via a remote alias, the remote alias must be located in the same `AlfabetMS.xml` file as the server alias for the Alfabet Web Application. This option is described in the section Copying Existing Alias

Configurations to a Separate AlfabetMS.xml File in Another Directory of the chapter Working with the Alfabet Administrator . Please note that the remote alias will be deprecated in the near future and should not be used for new installations.

Creating a Server Alias for the Alfabet Web Application

ⓘ For a new server alias configuration, many fields in the editor will be prefilled with best practice settings. This section is limited to a short description of the basic parameters not having any default value and necessary to set to run the system. Parameters that are required for special configuration scenarios are described in the chapter about the respective scenario. A detailed overview over all configuration parameters for the Alfabet platform is given in the section Configuration Attributes for the Alfabet Components.

1) Open the Alfabet Administrator from the Windows® Start Menu. For a new installation of Alfabet, an information message will be displayed that explains that no `AlfabetMS.xml` file is found. This message can be confirmed without further action required. The `Alfabetms.xml` file will be created automatically during the next steps.

2) In the explorer of the Alfabet Administrator click the **Alfabet Aliases** node and click **New** > **Create Alias** in the toolbar. An editor opens.

3) Edit the following fields:

   Overview tab

   - **Name**: The default value is displayed. If necessary, enter a unique name for the server alias.

      ⚠️
      - The name of the server alias must be unique. If your `AlfabetMS.xml` contains two server aliases with the same name, both configurations will be invalid.

      - It is recommended not to use special characters or spaces in alias names.

   - **Is Remote**: Leave the checkbox empty.

   - **Host**: The loop back address of the host is entered as the default. The default value should not be changed.

   - **Port**: The default value (port 1880) is displayed. If necessary, enter the port number of the communication port used to listen for incoming TCP connections.

   - **Server**: Alias name of the Alfabet component. The default value is displayed. If necessary, enter a unique name for the server alias.

      ⚠️
      - The name of the server alias must be unique. If your `AlfabetMS.xml` contains two server aliases with the same name, both configurations will be invalid.

      - It is recommended not to use special characters or spaces in alias names.

   - **Ensure Security**: If checked,.NET EnsureSecurity is used to secure the communication channel between the Alfabet components. The machines on which the Alfabet components are installed must be members of the same active directory.

> (i) Various mechanisms for communication between Alfabet components require that the Alfabet components are members of the same Microsoft® Active Directory® forest.
>
> If you want to work with components that are members of different Microsoft® Active Directory® forests or if you encounter problems when working with Alfabet components from different releases, please contact Software AG Support for a detailed description of the required configuration.

- **Help Server**: The URL to access to the Alfabet Online Help files. Enter the full Web server name and the name of the subdirectory in the application directory containing the Help files created during the basic installation. For a typical installation, this is the subdirectory *Help* of the application directory of the Alfabet Web Application specified in step 16 of the basic installation. If the Alfabet Online Help is installed without the Alfabet Web Application, this is the application directory for access to the help files.

  Server variables can be used to define part of the information as a variable with the value set in the **Variables** tab of the server alias configuration. For more information, see Defining Connections Based On Server Variables.

- **Web Server**: The base URL used for the creation of express views and object views. This setting is required to ensure full functionality of the Alfabet user interface. Enter the full Web server name and the name of the application directory created during the basic installation.

  Server variables can be used to define part of the information as a variable with the value set in the **Variables** tab of the server alias configuration. For more information, see Defining Connections Based On Server Variables.

- **Search Index Directory**: Insert the full directory name of the search index directory specified during the basic installation.

  Server variables can be used to define part of the information as a variable with the value set in the **Variables** tab of the server alias configuration. For more information, see Defining Connections Based On Server Variables.

Database Settings tab:

- **Database Driver**: Select the database server type in the drop-down field.

- **Database Driver Type**: If the **Database Driver** attribute is set to `SQL Server`, select one of the following:

  - `Microsoft.Net Framework` for the driver supporting .NET Framework only. This driver has been used for all Alfabet installations prior to release 10.9.0 as only supported driver.

  - `Microsoft Sql Client` for the driver supporting both .NET Framework and .NET Core. This is the default setting for new installations.

- **User Name**: Enter the user name specified for the connection to the database.

  > (i) For Oracle® databases the user name is identical to the schema name.

- **Password**: Enter the password specified for the connection to the database.

(i) User Name and Password are only required for standard login to the database. In case the Alfabet component will log in to a Microsoft® SQL Server® using Windows integrated login, no user name and password must be specified. For information about the different login mechanisms to the Alfabet database see Configuring Authentication Between the Alfabet Components and the Database Server.

- **Database**: Enter the name of the Alfabet database:

  - For a Microsoft® SQL Server® database, use the notation servername\databasename for a default instance or servername\instancename\databasename for a named instance.

  - For an indirect connection to an Oracle® database, use the Net Service Name of the database configured with the `Oracle® Net Assistant`.

  - For a direct connection to an Oracle® database this parameter is ignored.

- **Encrypt Connection**: For a connection to a Microsoft® SQL Server® only: Select the check box, if you want SSL/TLS to be applied to the connections between the Alfabet component and the Alfabet database.

- **Indirect Connection/Direct Connection**: For a connection to an Oracle® database only: Select whether a direct or indirect connection to the Oracle® database should be established.

  (i) For reasons of performance, a direct connection is recommended for connections to the Alfabet database. Indirect connections require the additional installation of a client application. Although some Oracle® features are only supported for indirect connections, none of these features are required to work with Alfabet. It is possible to change from a direct connection to an indirect connection if any of the restrictions for direct connections become critical in the future. The main restrictions of direct connections to Oracle® databases include:

    - Only the TCP/IP protocol is supported for the connection.

    - Advanced authentication such as OS authentication and proxy authentication are not supported.

    - Advanced configurable Oracle® functionalities such as Real Application Cluster, Oracle Loader, Transparent Application Failover, and Oracle Transaction Guard are not supported.

- **Database Host**: For a direct connection to an Oracle® database only. Enter the host name or IP address of the database server host.

- **Database Port**: For a direct connection to an Oracle® database only. Enter the port of the database server host the Alfabet Server should use for connection to the Alfabet database.

- **Database Service Name**: For a direct connection to an Oracle® database only. Enter the SID of the Alfabet database.

- **Clean invalid characters from strings during reading and writing**: Select the check box if you want strings to be checked for validity prior to check-in or when reading the string from the Alfabet database. For example, in Microsoft® Internet Explorer® 11, copying text from a PPT file to paste in a text field (such as a **Description** field) in an editor in Alfabet may result. Invalid characters such as non-permissible special characters not allowed are cleared and the string is written to the database and displayed in the Alfabet user interface without the special characters.

ⓘ This option can cause a loss of performance. It should only be checked if problems with special characters are detected.

Application Server tab: The settings in this tab define whether event queuing or.NET remoting is used for the execution of processes like batch tool or ADIF job execution..NET remoting will be deprecated in the near future. For a new installation, event queuing should be used. This requires the following settings:

- **Use Event Queue for All Jobs**: Select the checkbox.

ⓘ For a description of the other attributes that need to be set for.NET remoting, see the complete list of server alias attributes in the section Configuration Attributes for the Alfabet Components.

4) Configure the authentication mechanisms that shall apply to user login in the **Client Settings** tabs **Authentication**, **Actualization**, and **Authorization**. The authentication mechanisms that can be used with Alfabet and the required settings are described in detail in the section Configuring User Authentication.

5) Click **OK** to save the configuration. The server alias 🗄 is now displayed in the **Administrator** explorer.

Creating a Server Alias for the Alfabet Server

You can base the server alias of the Alfabet Server application on the server alias already available for the Alfabet Web Application accessing the same Alfabet database. It is required to create a separate server alias for the Alfabet Server with a different alias name.

In the tool Alfabet Administrator, do the following to create a server alias for the Alfabet Server as a copy of the server alias for the Alfabet Web Application:

1) In the explorer, expand **Alfabet Aliases**, right-click the server alias that you want to create a copy of and select **Create alias as Copy**.

2) In the editor that opens, change the information in the following fields of the **Overview** tab:

- **Name**: Enter a unique name for this server alias.

⚠️
- The name of the server alias must be unique. If your `AlfabetMS.xml` contains two server aliases with the same name, both configurations will be invalid.

- The name of the server alias should be short. The server alias name is used to create a path to a temporary directory for storing Alfabet Server related files during operation of the Alfabet Server. The length of the file names for the temporary files including the path information must not exceed 255 characters.

- It is recommended not to use special characters or spaces in alias names.

- **Server**: Change the Alias name of the Alfabet component.

⚠️ The alias name must be unique. If your `AlfabetMS.xml` contains two aliases with the same name, both configurations will be invalid.

- **Host**: Enter the IP address or DNS name of the Alfabet Server host. The loop back address of the server host is entered as the default.

  ℹ️ If the Alfabet Server and the Alfabet Web Application are installed on the same host, performance is best if the loop back address is used for the connections between the Alfabet Web Application and the Alfabet Server. Specification of the IP address or DNS name of the Alfabet Server as host address might lead to decreased performance.

- **Port**: The default value (port 1880) is displayed. If necessary, enter the port number of the communication port used by the Alfabet Server to listen for incoming TCP connections from the Alfabet Web Application.

- **Ensure Security**: If checked,.NET EnsureSecurity is used to secure the communication channel to the Alfabet Server. The machines on which the Alfabet components are installed must be members of the same active directory.

  ℹ️ Various mechanisms for communication between Alfabet platform components require that all Alfabet components are members of the same Microsoft® Active Directory® forest.

  If you want to work with components that are members of different Microsoft® Active Directory® forests or if you encounter problems when working with Alfabet components from different releases, please contact Software AG Support for a detailed description of the required configuration.

- **Search Index Directory**: Specify the full directory name of the search index directory specified during the basic installation.

- **SMTP Server**: Specify the fully qualified domain name or IP address of the SMTP server that is used to send emails.

  ℹ️ By default, the Alfabet Server tries to use a local SMTP server. Therefore, the configuration of the **SMTP Server** attribute is not required if you want the Alfabet Server to use an SMTP server that is installed on the Alfabet Server host.

3) Open the **Server Settings** tab and define the mode for the specification of the sender email address for emails sent by the system with the **System Sender Email Account** and **Failover Sender E-Mail Account** fields. For detailed information about the configuration steps required to send system emails in the scope of the Alfabet functionalities, see the section Specifying Sender Email Addresses.

4) Optionally, you can adapt the other parameters of the server alias to your requirements. For an overview of all available parameters, see Configuration Attributes for the Alfabet Components.

5) Click **OK** to save your changes.

Creating a Remote Alias

The following procedure is optional and only added here for backward compatibility reasons. Remote alias configurations will be deprecated in the near future and shall not be used for new installations.

A remote alias must only be created if the Alfabet Web Application is configured to execute processes via.NET remoting services. The remote alias must then be created for the Alfabet components, including the Alfabet Web Application, for accessing the Alfabet database via the Alfabet Server. The remote alias configuration includes the parameters for establishing the connection to the server and client specific settings. It is possible to specify multiple remote alias configurations for access to the same server alias but with different client settings.

Configuring the Alfabet Web Application to Hand Over Processes to the Alfabet Server for Execution

The Alfabet Server executes many processes triggered by users on the Alfabet user interface or by system administrators via batch processing tools. This includes sending of emails, ADIF job execution, starting of workflows and generation of the full text search index. Currently the handover of processes to the Alfabet Server for execution can be handled via two methods:

- **Event queueing**: Processes to be executed via the Alfabet Server are written into an event queue in the Alfabet database. The Alfabet Server scans the event queue in regular intervals and executes all processes that are pending. Multiple Alfabet Server s can process events simultaneously to enhance performance. This method is highly recommended because the method described below will be deprecated in the near future.

- **Remote service calls**: Processes are directly handed over to the Alfabet Server for execution on a direct connection established via a remote alias. This method will be deprecated in the near future on transition to using.NET Core. Remote service calls are not supported by.NET Core.

The methods are configured as follows:

To configure the Alfabet Web Application to queue processes in an event queue for processing by the Alfabet Server:

1) In the table, select the server alias of the Alfabet Web Application and click the **Edit** 🖉 button in the toolbar. An editor opens.

2) Go to the **Application Server** tab.

3) Select the **Use Event Queue for All Jobs** checkbox.

To configure the Alfabet Web Application to connect to the Alfabet Server via a direct connection, both a remote alias and a server alias must be available in the `AlfabetMS.xml` configuration file read by the Alfabet Web Application. the following configuration is required in the server alias of the Alfabet Web Application:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer and right-click the server alias for which you want to create a remote alias and select **Create Remote Alias**. The new remote alias is displayed in the explorer.

2) In the explorer, click the **Alfabet Aliases** node.

3) In the table, click the new remote alias and click the **Edit** button in the toolbar. An editor opens.

4) In the **Overview** tab, values are taken over automatically from the server alias. The following fields may require changes:

- **Host**: If `localhost` is specified as **Host** and the Alfabet component connecting to the Alfabet Server is located on a different host, change the **Host** to the IP address or DNS name of the Alfabet Server host.

- **Name**: By default, the name of the remote alias is set to <server alias name>-`remote`. Changing the name might be useful if you want to specify multiple remote alias configurations. It is recommended that you specify the remote alias name as <server alias name>-<purpose of remote alias>-Remote (For example, Alfabet-Testing-Remote). Alias names should be short and should not contain special characters.

The following settings must be identical to the settings on the server side and must not be changed:

- **Port**

- **Server**

- **Ensure Security**

5) Configure the authentication mechanisms that shall apply to user login with the remote alias in the **Client Settings** tabs **Authentication**, **Actualization**, and **Authorization**. The authentication mechanisms that can be used with Alfabet and the required settings are described in detail in the section Configuring User Authentication.

6) In the table, select the server alias of the Alfabet Web Application and click the **Edit** button in the toolbar. An editor opens.

7) Go to the **Application Server** tab.

8) Select the **Use Application Server and Net Remoting Service** checkbox.

9) Select the remote alias defined for the connection to the Alfabet Server in the **Remote Alias for Connection to the Application Server** field.

10) To implement the asynchronous export and import of data capture templates, select the **Use Server to Execute ADIF Jobs** checkbox. This checkbox can also be optionally activated for performance reasons to execute all ADIF jobs via the Alfabet Server. It is recommended to activate the checkbox.

11) Optionally, activate the optional features that can be executed by the Alfabet Server for performance reasons by selecting the checkbox:

- **Use Server to Execute Full-Text Search**

- **Use Server to Update Workflow Templates**

Setting up the Initial Alfabet Database

An Alfabet database that provides the initial state of Alfabet is delivered with the software as an ADBZ database archive file. This database archive file is required to initially populate the Alfabet database.

To read the ADBZ file to the Alfabet database:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer and right-click the server alias that connects to the Alfabet database and select **Restore Database Archive**.
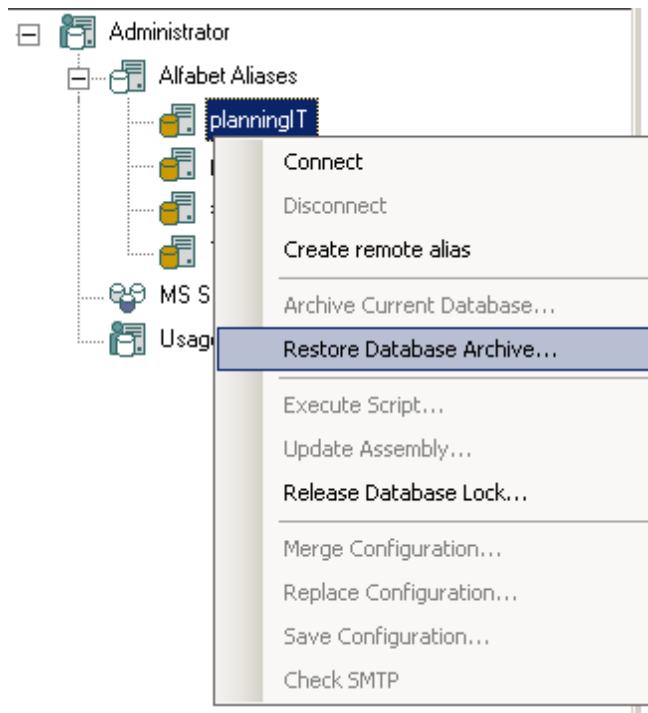
2) In the dialog box that opens, select the file name and location of the archive file that you want to restore.

> (i) Alfabet database archive files have the extension ADBZ or ADB.

3) The following options can be selected when restoring the Alfabet database:

- **Squeeze Audit Tables**: If the checkbox is selected, the audit tables are scanned for obsolete entries. Any entries that were generated during, for example, the batch update of data via batch utilities without documenting any audit relevant changes are deleted from the audit tables restored from the ADBZ file.

- **Ignore case sensitivity validation**: A case-sensitivity check has been implemented for restore of the Alfabet database from ADBZ file. If the target database is case-insensitive and the ADBZ file was created from a case-sensitive database, the restore process will not be performed and a message informs about the incompatibility of the databases. Select the checkbox to deactivate the check and to allow a case-sensitive database to be restored to a case-insensitive target database. The attribute is deselected by default.

4) Click **Restore** to start the restore process. It may take a few minutes before the restore process is completed.

5) In the **Info** window that opens, click **OK**.

Verifying the Connection to the Alfabet Database

Expand the **Alfabet Aliases** node in the **Administrator** explorer and right-click the server alias that shall connect to the Alfabet database and select **Connect**. A login window is displayed. Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database. If the connection is successful, the server alias is expanded, and the color of the server alias symbol will change to green. If the connection is not successful, an error message will appear.

Setting Up the Alfabet Web Application

The following information is available:

- Configuring the web.config Files for the Alfabet Web Application

  - Functionality-Related Settings for the Configuration of the Alfabet Web Application

- Configuring the Application for the Alfabet Web Application

- Performance Optimization for the Alfabet Web Application

  - Settings in the Registry of the Web Server Host

  - Settings in the machine.config File of the Microsoft® Internet Information Server® host

  - Settings in the web.config and Alfabet.config Files of the Alfabet Web Application

- Controlling Folder Permission Rights

Configuring the web.config Files for the Alfabet Web Application

The Alfabet Web Application must be configured with the following configuration files:

- `web.config` file

  The `web.config` configuration file must be located in the Alfabet Web Application directory specified in step 12 of the basic installation. This configuration file is the master configuration file. General ASP.NET web application settings are directly configured in this file. The file also contains links to two subordinate files that contain the Alfabet specific settings. The `web.config` file can be modified using a standard text editor.

- `Alfabet.config` file

  The `Alfabet.config` configuration file must be located in the `config` subdirectory of the Alfabet Web Application directory specified in step 12 of the basic installation. This configuration file is required to enable the Alfabet Web Application to connect to the Alfabet database. The `Alfabet.config` file can be modified using a text editor in the Alfabet Administrator.

- `AppSettings.config` file

  The `AppSettings.config` configuration file must be located in the `config` subdirectory of the Alfabet Web Application directory specified in step 12 of the basic installation. This configuration

file contains Alfabet-specific SAML configuration settings and is only relevant if SAML authentication is used. The `AppSettings.config` file can be modified using a standard text editor. For information about the configuration of SAML authentication, see [Configuring the Alfabet Web Application to Use Authentication via Federated Authentication](#).

The IIS_IUSRS (or the configured application pool user if application pool-specific accounts are configured on Internet Information Services®) must have write permissions for all files and folders in the physical directory of the Alfabet Web Application.

To create the required configuration files for your Alfabet Web Application, carry out the following steps in the Web Application directory:

1) Copy one of the configuration files from the `Example` subdirectory to the Alfabet Web Application directory. Which file must o be copied depends on the configuration mode that you would like to configure for the Alfabet Web Application:

   - For SAML authentication, use `Web.SAML2.config`.

   - For Windows® Single Sign-On authentication, use `Web.Windows.config`.

   - For all other authentication modes, use `web.default.config`.

2) Rename the file to `web.config`.

3) Copy the `config` subdirectory in the `Example` directory to the main directory of the Alfabet Web Application. This directory contains the subordinate configuration files `Alfabet.config` and `AppSettings.config`.

4) Open the Alfabet Administrator.

5) Click the **Alfabet Aliases** node in the explorer. A workspace with a toolbar opens.

6) In the toolbar, click **Tools** > **Configure Alfabet.config**. An editor opens.

7) Click the browse ▦ button on the right of the **Web Folder** field and select the main directory of the Alfabet Web Application in the directory browser. The `Alfabet.config` file in the ***config*** subdirectory of the selected directory opens in the editor.

8) Specify the path to the `AlfabetMS.xml` file and the server alias defined in the `AlfabetMS.xml` in the XML element `add` of the XML element `alfaSection`:

```
<alfaSection>

    <add key="msFile"
    value=" C:\programms\Alfabet\programms\AlfabetMS.xml "/>

    <add key="alias" value=" AlfabetViewer "/>

    ...

</alfaSection>
```

> ⓘ Please consider the following:
>
> - The specification of the server alias name is case-sensitive.
>
> - The `AlfabetMS.xml` file is located in the ***Programs*** subdirectory of the Alfabet installation directory.
>
> - For more information about configuring the server alias, see [Creating a Server Alias for the Alfabet Web Application](#).

9) Depending on the Alfabet features that you want to use and the authentication and security mechanisms implemented, additional settings are required in this or any of the other configuration files. Change the settings as needed. The most important settings for activation of features are listed below in the section Functionality-Related Settings for the Configuration of the Alfabet Web Application. Other optional settings are described in this reference manual in the context of the respective security feature which is implemented via the setting. For more information see Considering Security Issues.

10) Click **Save** to save the changes and close the editor.

Functionality-Related Settings for the Configuration of the Alfabet Web Application

The following settings might be required in the `web.config` file depending on the features that you want to use in Alfabet. Other settings might be required depending on your local network requirements. These settings are not listed here.

The settings are listed separately for each configuration file.

web.config settings:

- The cookie name configured in the session state settings must be unique for each application running on the server. Please note that the value for the XML attribute `cookieName` in the XML element `sessionState` in the `web.config` file must be specified differently for the `web.config` file used in the test environment and for the `web.config` file used in the production environment file. For example:

    ```
    <sessionState cookieName="Alfabet912" mode="InProc" />
    ```

- Make sure that the following setting is configured for the XML element `compilation` that is a child XML element of the XML element `system.web`. This setting needs to correctly point to.NET Framework 4.8:

    ```
    <compilation targetFramework="4.8" debug="false" defaultLanguage="cs">
    ```

- The default settings in the Internet Information Services are different depending on the update you have currently installed. If you see the error message `System.Web.HttpException: A potentially dangerous Request.Path value was detected from the client (&)` when rendering the Alfabet user interface, add the following attriubte to the XML element `httpRuntime` in the `web.config` file of the Alfabet Web Application: `requestPathInvalidCharacters=""`

- Alfabet views can be exported to different file formats using the **Export** button in the toolbar of the view. This feature is only available if the following element is added as child element of the XML element `handlers`:

    ```
    <add name="AlfaDownload" type="AlfabetWeb5.Common.FileExportHandler,
    AlfabetWeb5" verb="GET" path="FileExport" />
    ```

- To enable publication via the **Export** button in the toolbar of a view while single sign on mechanisms are used for authentication, the following code must be added as child element to the XML element `configuration`:

    ```
    <location path="PubSource.aspx">

        <system.web>

            <authorization>
    ```

```
            <allow users="*"/>

        </authorization>

    </system.web>

</location>
```

⚠ Please note that a configuration of the PubSource.aspx authentication mode on the Web server level is additionally required to publish data in single sign on mode. this configuration is described as part of the Web server setup in the section Configuring the Application for the Alfabet Web Application.

- If the Alfabet RESTful API shall be used, make sure that the XML element `handlers` has the following child elements apart from child elements that are already included for other processes:

  ```
  <remove name="ExtensionlessUrlHandler-Integrated-4.0" />

  <add name="AlfaRest1" type="AlfabetWeb5.api.v1.AlfaRestService,
  AlfabetWeb5" verb="*" path="api/v1" />

  <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*."
  verb="GET,HEAD,POST,DEBUG,PUT,DELETE"
  type="System.Web.Handlers.TransferRequestHandler"
  resourceType="Unspecified" requireAccess="Script"
  preCondition="integratedMode,runtimeVersionv4.0" responseBufferLimit="0"
  />
  ```

- If a reference error is displayed that states that the mainSlideInToolbar is not defined, the following child XML element of the XML element `<alfaSection>` in the `Alfabet.config` file of the Alfabet Web Application is missing and must be added to solve the issue:

  ```
  <add key="use_nc_url_param" value="true"/>
  ```

  This setting is included in the example files delivered with the current release. Customers not using the example files should make sure that the setting is available in their configuration.

- If configured reports are defined that are based on data cubes, the Web server must have access rights to the cube definitions on the Analytics Server. This can be achieved in one of the following ways:

  - Grant the web server access to the databases on the Analytics Server by changing the access rights on the Analytics server. This is the preferred method from a security perspective.

    ⓘ Web servers are usually running as user NETWORK SERVICE.

  - Configure the Web server to access the Analytics Server as a user already configured in the Analytics server to have access to the databases independent of the usual user settings for the Web server. This requires a change in the `web.config` file of the Alfabet Web Application. An XML element `identity` must be added as child to the XML element `system.web` as follows:

    ```
    <identity impersonate="true" userName="Name" password="password" />
    ```

    The XML attributes `userName` and `password` must specify a user name and password valid for access to the Analytics Server.

- If large files shall be uploaded to the Internal Document Selector available for Alfabet, ensure that the following settings are available in the `web.config` file:

```
<system.web>

    <httpRuntime appRequestQueueLimit="50000"
    maxRequestLength="2147483000" maxUrlLength="2097150"
    maxQueryStringLength="2097150" targetFramework="4.5" />

</system.web>

<system.webServer>

    <security>

        <requestFiltering>

            <requestLimits maxAllowedContentLength="2000000000" />

        </requestFiltering>

    </security>

</system.webServer>
```

- The ASP.NET version information is part of the HTTP header. To enhance security, this information can be removed from the HTTP header by specifying `false` for the XML attribute `enableVersionHeader` of the XML element `httpRunTime` in the `web.config` file:

```
<httpRunTime enableVersionHeader="false">
```

- For security reasons, the following section is available in the XML element `system.webServer` of the `web.config` file provided with Alfabet. It is recommended that this section is included in the `web.config` file:

```
<httpProtocol>

    <customHeaders>

        <add name="Cache-Control" value="no-cache, no-store, must-
        revalidate, pre-check=0, post-check=0, max-age=0, s-maxage=0" />

        <add name="Pragma" value="no-cache" />

        <add name="Expires" value="-1" />

        <add name="Strict-Transport-Security" value="max-age=31536000;
        includeSubDomains" />

        <remove name="X-Powered-By" />

        <add name="X-Frame-Options" value="SAMEORIGIN" />

        <add name="X-XSS-Protection" value="1; mode=block" />

        <add name="X-Content-Type-Options" value="nosniff " />

        <add name="Referrer-Policy" value="same-origin, strict-origin-
        when-cross-origin"/>

        <add name="Content-Security-Policy" value="default-src 'self'
        'unsafe-eval' 'unsafe-inline' *.stt.speech.microsoft.com; img-src
        'self' data:; " />

        <add name="Permissions-Policy" value="geolocation=()"/>

    </customHeaders>

</httpProtocol>
```

⚠ The `Content-Security-Policy` settings disable the display of images or media from other web servers. If you would like to display content from other web servers in the Alfabet user interface such as videos embedded in automated assistants or images

embedded in HTML text, you must extend the `Content-Security-Policy` definition with the specification of the image or media source. For more information, see the documentation of the `Content-Security-Policy` HTTP header at, for example, https://content-security-policy.com/.

- The following section is part of the standard web.config delivered with the application. It ensures that the Alfabet internal error message will be displayed if system errors occure. It is recommended to use the Alfabet internal error message instead of the standard web server error message.

Alfabet.config settings:

- A security mechanism has been implemented that ensures that all requests sent to the Alfabet Web Application are sent from the same client browser. If the client host check is not positive, the session will be terminated, and the login screen will be displayed so that the user can login again. Please note that this security check takes network parameters into account. For example, changing from LAN to WLAN during a session will terminate the session. The security check for IP context validation can be deactivated in the `Alfabet.config` file of the Alfabet Web Application. You should only deactivate the check if your enterprise encounters the frequent termination of active sessions by adding the following code as a child element to the XML element `AlfaSection`:

```
<add key="ValidateSessionContext" value="false"/>
```

> The example, `Alfabet.config` files delivered with the installation already contain the code but are commented out and thus inactivated.

- The embedded third-party component Essential Objects® is used to enhance the image quality for images exported to Microsoft Word® or PowerPoint® files as well as to activate export options for these formats for some configured graphic reports such as branching diagram reports and gallery reports. It requires activation in the `Alfabet.config` file of the Alfabet Web Application by adding the following to the XML element `alfaSection`:

```
<add key="eo_publishing" value="on" />
```

- When the information about the server alias and user profile of the current session is very long, the display of this information in the main menu bar of the Alfabet user interface can lead to problems with the display of menu buttons on small screens. Therefore, it is possible to move the information about the server alias and user profile to the **Help** menu in a sub-menu option **About Current Session**. To display the information in the sub-menu **About Current Session** in the **Help** menu, the following `key` entry must be added in the XML element `AlfaSection`:

```
<add key="cur_session_info_position" value="help_menu" />
```

If the value attribute is set to `main_menu` or the entry is not added to the `Alfabet.config` file, the information will be displayed in the main menu.

- An anti-virus check for document is implemented with the following keys in the `Alfabet.config` file, which are set to `true` by default:

  - Anti-virus check of documents on upload to the **Internal Document Selector** is activated with the `scan_malware` key:

```
<add key="scan_malware" value="true"/>
```

  - Anti-virus check of Microsoft® Word® and PowerPointMicrosoft® PowerPoint® templates prior to each publication process via the **Publications** functionality is activated with the `scan_templates_for_malware_when_publishing` key:

```
<add key="scan_templates_for_malware_when_publishing" value="true"/>
```

The scan will be performed with the anti-virus scanner implemented on the server hosting the Alfabet Web Application. If no anti-virus scanner is explicitly implemented, the anti-virus capabilities of the underlying Windows® operating system will be used to scan document during upload.

- For test and configuration environments, the stack trace information can be added to the error messages displayed in the Alfabet components. It is recommended that the Alfabet components are configured to not display this detailed information in the production environment. To display the call stack in the error message, the following entry must be added to the XML attribute `key` in the XML element `AlfaSection`:

  ```
  <add key="debug_mode" value="true"/>
  ```

- If you have installed a REST client that will send a high number of service calls per second to the Alfabet RESTful service API, it might be required to increase the maximum allowed number of requests per second. The processing of incoming RESTful service calls is limited to 300 per second per default. Change the XML attribute `value` as needed:

  ```
  <add key="max_api_requests_per_second" value="300"/>
  ```

- For RESTful service calls to the Alfabet RESTful service, API security for data transmission can be enhanced optionally by specification of the JSON Web Token (JWT) for sending JSON objects via the Alfabet RESTful service API. Per default, the JWT is hard-coded and therefore the same for all Alfabet installations. To change the JWT for an installation, an individual JWT must be base64 encrypted and the encrypted version must be entered in the in the `Alfabet.config` file of the Alfabet Web Application. The following XML element must be added as child element of the XML element `alfaSection` to define the token:

  ```
  <add key="ApiJwtBase64Key" value="{Base 64 Encrypted key}"/>
  ```

- Memory consumption of running Alfabet Expand Web sessions and sub-sessions thereof is high because of the complexity of the configuration data. The maximum number of sessions that can be run in parallel is therefore restricted to 5 parallel sessions to avoid performance problems. This also includes sub-sessions. If the maximum number of (sub) sessions is reached, an attempt to open a new session will fail and a message will be displayed that informs about the exceeded session limit.

  Please note that a session is persistent for 20 minutes if the user terminates the session without logout. A server restart is required to terminate the sessions earlier.

  If problems with exceeding numbers of parallel Alfabet Expand Web sessions occure frequently, the maximum number of allowed parallel sessions can be changed from 5 to any other value with the following child element of the XML element `alfaSection`:

  ```
  <add key="expandwebsessionlimit" value="5"/>
  ```

Configuring the Application for the Alfabet Web Application

ⓘ Although multiple virtual directories can be deployed on the same Web site, this is not recommended for the productive instance of Alfabet. At a minimum, the different instances must be deployed in different application pools. Furthermore, running multiple instances of the Alfabet Web Application concurrently results in increased memory requirements. The minimum and recommended memory size documented for the Web Application Server in the technical

requirements must be available for each of the instances deployed on the same physical machine. For more information, see the section in the *Technical Requirements*.

> (i) The user running the Web server processes must have Read/Write permissions on the directory containing the Alfabet Web Application and its subdirectories.

The application directory for the Alfabet Web Application is created automatically during the installation of the Web Application. The name of the application pool is <name of application directory>Pool.

If the application pool was not created during installation, it must be added manually after installation as described below.

To create an application pool on the Internet Information Services® Web server and configure the Alfabet Web Application to use it:

1) Open the Internet Information Services® Manager.

2) Expand the web server node in the explorer and click on the sub-node **Application Pools**.

3) In the **Actions** pane on the right, select **Add Application Pool**.

4) In the dialog box that opens, define the following fields and then click **OK**:

   • **Name**: Enter a name for the new application pool.

   • **.NET Framework version**: Choose version 4.0.30319.

   • **Managed Pipeline Mode**: Select Integrated.

5) Click **OK** to save your settings.

6) In the table in the center pane, click the new application pool and select **Advanced Settings** in the **Actions** pane on the right.

7) In the dialog box that opens, go to the **Process Model** section and set **Idle Time-out (minutes)** to 0.

8) Go to the **Recycling** section and set **Regular Time Interval (minutes)** to 0.

> (i) The application pool is represented by a worker process (as w3wp.exe visible in the Task Manager). This worker process is started on first access to the Web Application in the application pool. To make sure that potential errors in the Web Application do not affect too many requests, the Internet Information Services® can be configured to recycle the worker process. Recycling is done by starting a new worker process after a configurable amount of time. The re-initialization of the web application can take up to 1 minute. It is therefore recommended t that you disable the automatic recycling of the worker process.
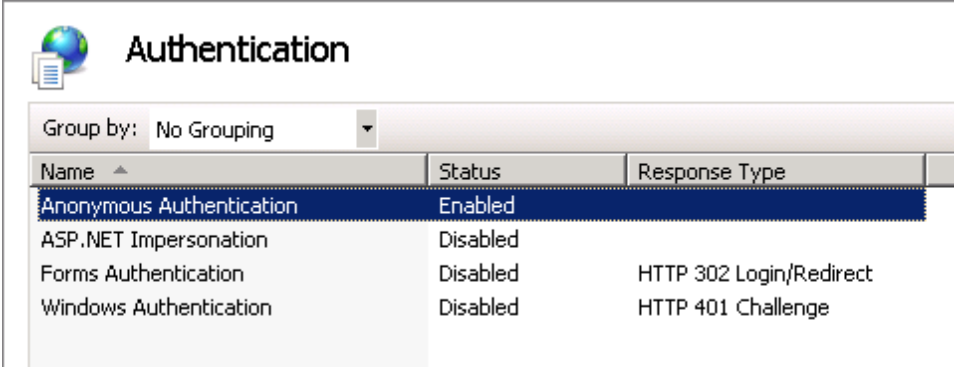
9) If your Alfabet database is located on a Microsoft® SQL Server® and you want Windows Sign-on to be used for the connection between the Alfabet Web Application and the Alfabet database, go to the **Process Model** section and set **Identity** to the service account used to access the Alfabet database.

> ⚠ The default language culture for the Alfabet user interface is identical to the culture settings implemented in the operating system for the user defined as **Identity** user. Therefore, a change of the Identity may lead to a change to the default language of the user interface.

10) Click **OK** to save your changes.

11) In the explorer of the Internet Information Services® Manager, select the application directory for the Alfabet Web Application. It is located under **Sites** > **Default Web Site**.

12) Select **Advanced Settings** in the menu on the right.

13) In the dialog box that opened, make sure the application is pointing to the correct application pool.

14) Click **OK** to save your changes.

15) In the section **IIS** in the center pane, double-click **Authentication**. The status of the available authentication modes is displayed.

16) Make sure that the authentication status is set according to the selected authentication method for access to Alfabet. To change the status of an authentication mode, right-click the status and select the required option in the context menu. The following settings are required:

   • Windows sign-on: **Windows Authentication** must be set to **Enabled** and **Anonymous Authentication** must be set to **Disabled**.

   • Standard login: **Windows Authentication** must be set to **Disabled** and **Anonymous Authentication** must be **Enabled**.

17) In the explorer, click the application directory of the Alfabet Web Application.

18) In the section **IIS** in the center pane, double-click **MIME Types**. A list of mime types is displayed.

19) If Alfabet Expand Web shall be used and AMM files should be created with Alfabet Expand Web, click **Add** in the **Actions** pane to add a new mime type. A new window opens. Enter the following in the fields and click **OK** to save the mime type:

   • **File name extension**:.amm

   • **MIME type**: application/Alfabet Meta-Model

20) In the explorer, select the application directory for the Alfabet Web Application.

21) In the **ASP.NET** section in the center pane, double-click **Session State**. The session state settings open.

22) Select the check box **In Process**.

23) Click **OK** to save your settings.

24) If you are using Windows sign on for authentication, do the following:

   1) Click the application directory of the Alfabet Web Application in the explorer.

   2) Click the **Content View** button in the bottom area next to the explorer:

   3) In the list, right-click PubSource.aspx and select **Switch to Features View** in the context menu.

4) Check that the title of the working area that opens is **PubSource.aspx Home**.

5) Double-click **Authentication**.

6) Enable **Anonymous Authentication** and disable **Windows Authentication**. To change the status of an authentication mode, right-click the status and select the required option in the context menu.

7) In the explorer, select the root directory of the web server.

8) In the **Actions** pane, click **Restart**.

9) Close the Internet Information Services® Manager.

Performance Optimization for the Alfabet Web Application

When accessing Alfabet with Alfabet Web Client s that are located in a wide area network, optimal performance requires the following configuration of the Alfabet Web Application as well as the Microsoft® Internet Information Server® hosting the Alfabet Web Application. A restart of the Web server host is required after the configuration.

Settings in the Registry of the Web Server Host

Set or edit the following `DWORD` values:

Key	DWORD value name	value (to be specified either hexadecimal or decimal)
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControl-Set\Services\HTTP\Parameters`	`MaxConnec-tions`	00002000 (Hexadecimal) 8192 (Decimal)

Settings in the machine.config File of the Microsoft® Internet Information Server® host

1) Using a text editor, open the file
   `%SystemRoot%\Microsoft.NET\Framework\v4.0.30319\CONFIG\machine.config.`

   > ⓘ For 64-bit operating systems, the path is:
   >
   > `%SystemRoot%\Microsoft.NET\Framework64\v4.0.30319\CONFIG\machine.config`

   > ⓘ The path information does not change with an update to.NET Framework v. 4.6..NET Framework v. 4.6 overwrites an existing version 4.0.

2) In the section **processModel** change the value for **allowDefinition** to **MachineToApplication**.

Settings in the web.config and Alfabet.config Files of the Alfabet Web Application

1) Using a text editor, open the web.config file located in the installation directory of the Alfabet Web Application.

2) Add the following as a child of the XML element `configuration`:

```
<system.net>

    <connectionManagement>

        <add address="*" maxconnection="1000" />

    </connectionManagement>

</system.net>
```

3) Change or, if the elements do not exist, add the following elements as child elements of the XML element `system.web`:

```
<httpRuntime enable="true" maxRequestLength="50000" minFreeThreads="88"

minLocalRequestFreeThreads="76" requestLengthDiskThreshold="88" />

<processModel enable="true" webGarden="false" maxIoThreads="100"
maxWorkerThreads="100"

autoConfig="false" shutdownTimeout="00:00:05" logLevel="Errors"/>
```

4) To improve the transfer performance of the website, the following code should be available. Modify the existing settings as needed:

```
<system.webServer>

    <httpProtocol>

        <customHeaders>

            [...]

            <add name="ETag" value=" " />

        </customHeaders>

    </httpProtocol>

    <directoryBrowse enabled="false" />

    <staticContent>

        [...]

        <clientCache cacheControlMode="UseExpires"
        cacheControlMaxAge="05:00:00" httpExpires="Fri, 01 Nov 2024
        00:00:00 GMT" setEtag="false" />

    </staticContent>

    <httpCompression sendCacheHeaders="true" />

</system.webServer>

<location path="favicon.ico">

    <system.webServer>

        <staticContent>

            <clientCache cacheControlMode="UseExpires"
            cacheControlMaxAge="05:00:00" httpExpires="Fri, 01 Nov 2024
            00:00:00 GMT" />
```

```
            </staticContent>

         </system.webServer>

      </location>
```

To further improve transfer performance, you can exclude `nc` parameters for URLs:

1) Open the Alfabet Administrator.

2) Click the **Alfabet Aliases** node in the explorer. A workspace with a toolbar opens.

3) In the toolbar, click **Tools** > **Configure Alfabet.config**. An editor opens.

4) Click the Browse ![button] button on the right of the **Web Folder** field and select the main directory of the Alfabet Web Application from the directory browser. The `Alfabet.config` file in the `config` subdirectory of the selected directory opens in the editor.

5) Add the following as child XML element to the `alfaSection` element:

   ```
   <add key="use_nc_url_param" value="false" />
   ```

   If the path is not specified with this setting, the default path to the `config` subdirectory of the Alfabet Web Application is used for SAML configuration.

6) Click **Save**. The change is saved and the editor is closed.

Controlling Folder Permission Rights

During installation, all folder rights required to run the Alfabet Web Application are set automatically. Nevertheless, your installation may require that access permissions are manually set. It is recommended that you check whether the following access permissions are set for the Microsoft® Internet Information Services® and the users accessing the web application via the Alfabet Web Client:

- Alfabet Web Application directory in the installation directory of the Alfabet components:

  - Read

  - Execute

- Runtime directory of the Alfabet Web Application directory:

  - Read

  - Execute

  - Modify

  - Write

- AlfabetMS.xml file of the Alfabet Server:

  - Read

  - Execute

  The account of the Microsoft® Internet Information Services® that requires access is:

- IIS_IUSRS (or the configured application pool user if application pool specific accounts are configured on your Internet Information Services®)

The specification of access permissions for Alfabet users depend on the configuration of access permissions in the Internet Information Services®:

- If Windows authentication is used, access permissions must be granted to all users that connect to the web application.

- If anonymous access is used, access permissions must be granted to the user account specified for anonymous access. By default, Microsoft® Internet Information Services® 7 and higher use IURS as account for anonymous access.

Running the Alfabet Server

The Alfabet Server will be installed automatically together with the Alfabet components. The `Alfaserver.exe` is located in the Alfabet Programs directory and can be accessed in the Windows® **Start** menu of the Alfabet components host.

You can either start the Alfabet Server as an application or configure the Alfabet Server to run as a Windows® service. For a production environment, the Alfabet Server should be implemented to run as a Windows® service independent of any user session.

- [Starting the Alfabet Server Application](#)

- [Configuring the Alfabet Server to Run as a Windows Service](#)

Starting the Alfabet Server Application

The Alfabet Server can be started during a user session as an application (alfaServer.exe). This mode is for ad-hoc and test operation. For a production environment, the Alfabet Server should be implemented to run as a Windows® service.

You will need an Alfabet user name and password to start the Alfabet Server as an application. For information about the configuration of a user with user name and password, see [Configuring User Authentication](#).

Administrator rights are required to start the Alfabet Server application.

To start the Alfabet Server for testing purposes:

1) In the Alfabet programs directory, double-click `AlfaServer.exe`. The **Alfabet Application Server** Window opens.

2) In the menu of the Alfabet Server, click **File** > **Start**.

3) In the **Login** window that opens, select the relevant server alias connecting to your test database and enter your user name and password.

4) Click **OK** to start the Alfabet Server.

Configuring the Alfabet Server to Run as a Windows Service

For a production environment, the Alfabet Server should be implemented to run as a Windows® service independent of any user session. The basic installation of Alfabet does not include the creation of this service.

In order to install the Alfabet Server as a service, you must be logged in as a local administrator.

⚠️ The Alfabet Server Service requires interaction with the desktop for technical reasons. This may lead to service shutdown when using a remote desktop session to the server host. If your Alfabet Server Service shuts down when disconnecting from a remote desktop session, use Windows Remote Management or remote shell to start the Alfabet Server Service.

To create an Alfabet Server service:

1) Open a command prompt as administrator on the Alfabet Server host.

   ⓘ To open a command prompt on Windows® Server 2012, right-click the **Start** ⊞ icon that appears when you move the mouse to the lower left corner and select **Command Prompt (Admin)**.

2) Go to the *Programs* subdirectory of the Alfabet Server 's root installation directory.

3) Run `AlfaServerServiceGenerator.exe` with the command line options listed below. The command must at least be run as:

   ```
   AlfaServerServiceGenerator.exe-server <server_alias_name>
   ```

This program will create an executable file that can be installed as a service. The resulting service name is case-sensitive.

The following command line options can be used with this command. The command line options are case sensitive.

Parameter	Meaning	Use	Default Value
`-server <server_alias_name>`	Server alias in the file `AlfabetMS.xml`. Please note that this name is case sensitive and must be used with the correct case here.	Mandatory	
`-service <service_name_windows>`	Windows® service name that is used in the registry	Optional	`"AlfaSrv"` + server_alias_name
`-displayservice<service_display_name>`	Displayed name in the Microsoft® Service Manager	Optional	`"Alfabet Server Service"` + server_alias_name

Parameter	Meaning	Use	Default Value
`-exe <ser-vice_exe_name>`	Name of the executable file for the service	Op-tional	`"AlfaServerService"` + server_alias_name + `".exe"`
`-startmode Manual | Automatic`	Start type for the service	Op-tional	manual
`-h` `-help`	Displays the list of possible pa-rameters	Alter-native	

⚠ Note the following when generating a server service:

- Whitespaces are not allowed in the command line options of the *AlfaServerServiceGenerator*.

- To ensure that valid Windows® executable names are generated through this process, the following restriction applies:

  - The value provided for the parameter`-exe` (or if this is not defined the server alias name in `AlfabetMS.xml`) must be alphanumeric.

    The only exceptions are the special characters '_' and "-". The administrator will be prompted with an error message if an invalid file name results from the server service generation.

4) Run the InstallUtil of the.NET framework on the alfa Server service executable. The following example uses the Microsoft® standard path to the utility on a 32-bit operating system:

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\Installutil.exe <Alfabet
Server Service executable file name>
```

and on a 64-bit operating system:

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Installutil.exe
<Alfabet Server Service executable file name>
```

The new service is now visible and can be started. It can also be further configured in the operating system console **Services**.

The Alfabet Server service generates entries in the Windows® application log. The logging includes error messages shown to users on the Alfabet interface. Depending on the severity of the error, the messages will be logged as either as an error, warning, or information.

⚠ Please note the following for the Alfabet Server service:

- You have to uninstall the service before updating Alfabet to a new release or patch release. For more information, see the documentation of the upgrade process.

- The Alfabet Server Service will stop if the connection to the Alfabet database is lost. To recover the service, manual action is necessary, or a script must be generated and scheduled for monitoring and restarting.

- In case the environment setup requires two Alfabet Server services to run on the same host, you must use the tool `AlfaServerServiceGenerator.exe` to generate additional services.

Testing the Installation

1) Open the browser.

2) Enter the URL of the Alfabet Web Application (for example `http://<full server name>/` Alfabet).

3) Click **OK** to connect.

4) Login to Alfabet (if enterprise authentication is not used).

Analyzing Performance with the Alfabet Platform Analyzer

The Alfabet Platform Analyzer (ppa.aspx) is a Web application that helps you identify the cause of performance problems if the Alfabet Web Client is not working as expected for the use

The performance of the Alfabet Web Client has been tested in Software AG test environments with good results. Please note that in a real-world scenario, a wide variety of platform parameters including, for example, virus scans and extensive CPU or memory usage by other processes may lead to a decrease in performance.

The Alfabet Platform Analyzer simulates an active Alfabet Web Client without affecting customer data in the Alfabet database and generates a report about the platform parameters that might be relevant for performance issues. The platform analysis report includes parameters that typically influence performance based on a scenario whereby access permissions exist that allow the relevant processes to retrieve the customer data. The report serves as a starting point to find the source of performance problems and will typically by required if a service request is submitted to Software AG 's Empower eService via https://empower.softwareag.com.

For security reasons, the Alfabet Platform Analyzer (`ppa.aspx`) requires login.

To analyze the platform and display test results, start the Alfabet Platform Analyzer `ppa.aspx` on the client host in order to generate the platform analysis report. The web address is <path to the Alfabet Web Application >/`ppa.aspx`. Login is required.

ⓘ Please note that if you start ppa.aspx multiple times, the performance measured in the first run might be lower than the performance in the repeated measurements because the browser used may cache the content used for testing.

Installation of Tools for Configuration and Administration in Alfabet

The tools available for purposes of configuration and administration of Alfabet are installed automatically during the installation process if the license key includes them. Some of the tools require a server alias to work with and/or special configurations in the web.config files of the Alfabet Web Application. This section describes in detail which configuration steps are required for the implementation of the individual tools. In general, the following configuration is required:

- The tool Alfabet Administrator does not require a server alias configuration. It directly connects to the Alfabet database with any of the server alias configurations included in the `AlfabetMS.xml` configuration file managed by the tool.

- The tool Alfabet Expand Windows directly connects to the Alfabet database with a server alias configuration. It can access the Alfabet database in parallel with the Alfabet Web Application.

- The Web version of Alfabet Expand and the Guide Pages Designer are part of the Alfabet Web Application for Cloud customers. They can be accessed via a web browser. They are using the server alias configuration of the Alfabet Web Application and can be run in parallel with the Alfabet user interface.

- Batch processing applications that trigger specific functionalities either connect directly to the Alfabet database with a server alias configuration or via remote access to a running Alfabet Server. It is recommended that these tools are run in remote mode. For more information about the tools that are available and the functionality requiring execution of a batch tool, see Configuring and Activating Alfabet Functionalities.
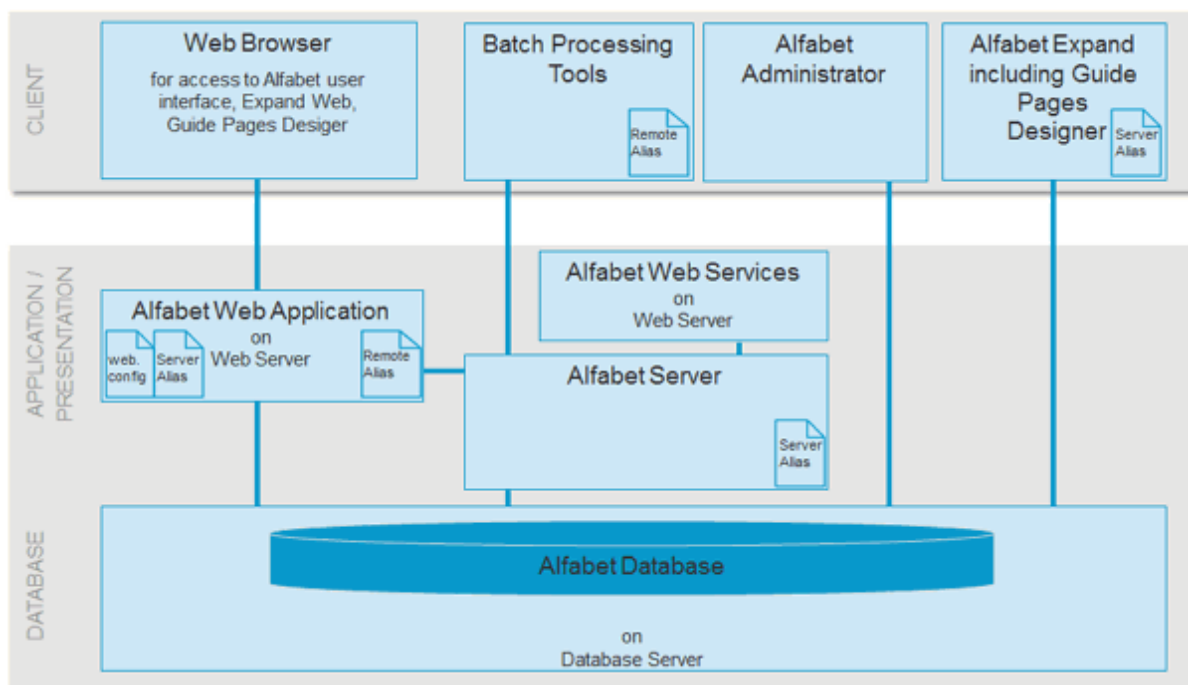
FIGURE: Overview of alias configurations and web.config file configurations for the installation of the Alfabet components

Each instance of any Alfabet component connecting to the Alfabet database must use a different server alias configuration with a unique name and server name setting. All other settings may be identical. In the Alfabet Administrator you can create a server alias as copy from another server alias and change the name

and server name setting to create a server alias with the same connection and behavior for each individual tool.

The following information is available about the configuration of the different Alfabet components for installation and configuration:

- Installation and Configuration of the Alfabet Administrator

- Installation and Configuration of Alfabet Expand and the Guide Pages Designer

  - Configuring Access to Alfabet Expand Windows and the Guide Pages Designer

  - Configuring Alfabet Expand to Update the Configuration from a Master Database via a Direct Connection

  - Configuring Alfabet Expand Windows to Scan the Meta-Model for Changes

- Installation and Configuration of Command Line Tools

- Special Configuration of Testing Environments

  - Changing the Session Timeout

  - Defining a Default Login User

  - Re-Routing Emails to a Defined Address for Testing

  - Rendering the Alfabet User Interface in Design Mode

Installation and Configuration of the Alfabet Administrator

The Alfabet Administrator allows system administrators to configure the Alfabet components and to perform database-related tasks such as restoring databases and updating the meta-model.

The Alfabet Administrator will be installed automatically together with the Alfabet components. The `AlfaAdministrator.exe` is located in the Alfabet Programs directory and can be accessed in the Windows® **Start** menu of the Alfabet components host.

No further configuration is required.

For an overview of the functionalities available via the Alfabet Administrator and a detailed description about how to work with the Alfabet Administrator, see Working with the Alfabet Administrator .

Installation and Configuration of Alfabet Expand and the Guide Pages Designer

Alfabet Expand is the main configuration tool for customizing the Alfabet solution and includes the *Designing Guide Pages for Alfabet* for the configuration of customized navigation pages. For detailed information about the configuration of navigation pages, see the reference manual *Designing Guide Pages for Alfabet*. For detailed information about the configuration of the Alfabet solution with Alfabet Expand, see the reference manual *Configuring Alfabet with Alfabet Expand*.

Alfabet Expand is available either Web-based or as an application to meet the requirements of different user groups:

- The Alfabet Expand application offers the complete range of configuration functionalities and is the tool for solution designers configuring an Alfabet solution installed at the customer. For detailed information about the configuration of the Alfabet solution with the Alfabet Expand application, see the reference manual *Configuring Alfabet with Alfabet Expand*.

- Alfabet Expand Web offers most common configuration functionalities for solution designers that do not have access to a local installation of the Alfabet components installed for their Alfabet solution (for example, when using an Alfabet Cloud solution). The Alfabet Expand Web configuration is not relevant for the installation of the Alfabet components at the customer.

It is possible to use multiple instances of Alfabet Expand in parallel. For example, a solution designer can configure guide pages with the Guide Pages Designer while another solution designer configures Alfabet functionalities using the Alfabet Expand application.

Nevertheless, it is recommended that only one instance of Alfabet Expand is used at a time. Otherwise changes made by one solution designer might be overwritten by another solution designer that concurrently works with another instance of Alfabet Expand. Although mechanisms can be implemented to limit the damage that can be done by concurrently working with multiple instances of Alfabet Expand, it cannot be guaranteed that changes made in one instance are completely honored in concurrently used instances of Alfabet Expand. For detailed information, see Configuring Alfabet Expand Windows to Scan the Meta-Model for Changes..

The following information is available:

- Configuring Access to Alfabet Expand Windows and the Guide Pages Designer

- Configuring Alfabet Expand to Update the Configuration from a Master Database via a Direct Connection

- Configuring Alfabet Expand Windows to Scan the Meta-Model for Changes

Configuring Access to Alfabet Expand Windows and the Guide Pages Designer

If your serial number includes access to the Alfabet Expand application, the application will be installed automatically together with the Alfabet components. The `AlfaExpand.exe` is located in the Alfabet Programs directory and can be accessed in the Windows® **Start** menu of the Alfabet components host.

The application directly accesses the Alfabet database and requires a server alias configuration to be available in the working directory. You can base the server alias of the Alfabet Expand application on the server alias already available for the Alfabet Web Application accessing the same Alfabet database. Nevertheless, it is required to create a separate server alias for the Alfabet Expand application.

In the tool Alfabet Expand, do the following to create a server alias for Alfabet Expand as a copy of the server alias for the Alfabet Web Application:

1) In the explorer, expand the **Alfabet Aliases** node, right-click the server alias that you want to create a copy of and select **Create the Alias as Copy**.

2) In the editor that opens, change the information in the following fields of the **Overview** tab:

- **Name**: Enter a unique name for this server alias.

⚠️
- The name of the server alias must be unique. If your `AlfabetMS.xml` file contains two server aliases with the same name, both configurations will be invalid.

- The name of the server alias should be short. The server alias name is used to create a path to a temporary directory for storing Alfabet Server related files during operation of the Alfabet Server. The length of the file names for the temporary files including the path information must not exceed 255 characters.

- It is recommended not to use special characters or spaces in alias names.

- **Server**: Change the Alias name of the Alfabet component.

⚠️ The alias name must be unique. If your `AlfabetMS.xml` file contains two aliases with the same name, both configurations will be invalid.

3) If you have licensed the Guide Pages Designer, the license key will be automatically added to the **Licenses** tab upon opening the server alias editor and stored when closing the editor via the **OK** button.

4) If multiple solution designers are configuring your Alfabet solution, go to the **Expand** tab and set the following parameters:

- Select the **Track Meta-Model Changes** checkbox.

- Optionally, change the period for tracking changes via the **Tracking Period** parameter. The default is one second.

ⓘ For more information about the tracking meta-model changes and the requirements to concurrently work with multiple instances of Alfabet Expand, see Configuring Alfabet Expand Windows to Scan the Meta-Model for Changes.

5) You need access to a running Alfabet Web Application, for opening the Alfabet user interface for testing and review functionalities.

It requires the following settings in the **Expand** tab of the server alias:

- **Test Web Application**: Enter the URL of the Alfabet Web Application that shall be used to open the Alfabet user interface from within the Alfabet Expand Windows application. The path must start with http:// or https://.

👁 For example:

```
http://hostname/AlfabetWebApplication
```

ⓘ Server variables can be used to define part of the information as a variable with the value set in the **Variables** tab of the server alias configuration. For more information, see Defining Connections Based On Server Variables.

6) Optionally, you can adapt the other parameters of the server alias to the special requirements of your solution designer. For an overview of all available parameters see Configuration Attributes for the Alfabet Components.

7) Click **OK** to save your changes.

After having configured the Alfabet Expand applications, named Alfabet users with access to the Alfabet Expand host can access the application if the check box **Has Access to Alfabet Expand** is activated in the user configuration of the user in the Alfabet database. For more information about user configuration, see *Defining and Managing Users* in the reference manual *User and Solution Administration*.

If the Alfabet Web Application that shall be used to open the Alfabet user interface from within the Alfabet Expand Windows application is configured to a use single sign on mechanism like Windows authentication or SAML authentication for user login, a special configuration is required on the Alfabet Web Application to accept the login from Alfabet Expand Windows:

1) Open the `web.config` file in the physical working directory of the Alfabet Web Application in a text editor. Make sure that the following code is available as child of the XML element `configuration`:

```
<location path="DesignTest.aspx">

    <system.web>

        <authorization>

            <allow users="*"/>

        </authorization>

    </system.web>

</location>

<location path="DesignTestSys.aspx">

    <system.web>

        <authorization>

            <allow users="*"/>

        </authorization>

    </system.web>

</location>
```

2) Open the Internet Information Services® Manager and click the Alfabet Web Application in the explorer.

3) Click the **Content View** button in the bottom area next to the explorer:
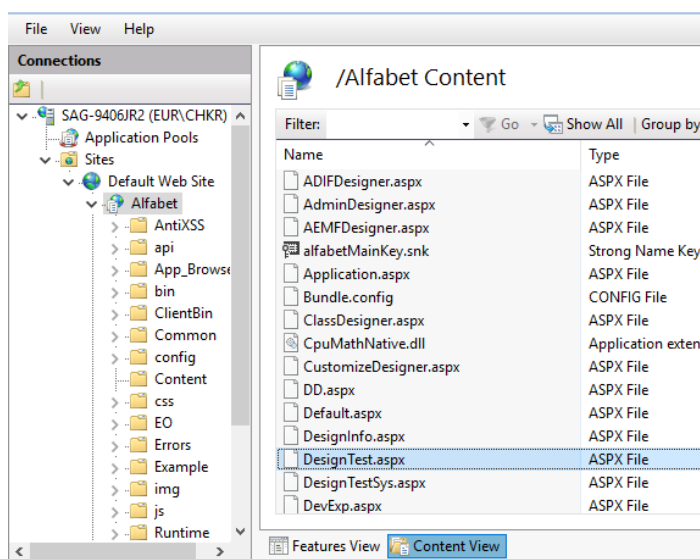
4) In the list, right-click `DesignTest.aspx` and select **Switch to Features View** in the context menu.

5) Check that the title of the working area that opens is **DesignTest.aspx Home**.

6) Double-click **Authentication**.

7) Enable **Anonymous Authentication** and disable **Windows Authentication**.

> (i) To change the authentication status, right-click the authentication mode in the table and select **Enable** or **Disable** in the context menu respectively.

Name	Status
Anonymous Authentication	Disabled
ASP.NET Impersonation	Enable

8) Click the Alfabet Web Application in the explorer.

9) Click the **Content View** button in the bottom area next to the explorer:

10) In the list, right-click `DesignTestSys.aspx` and select **Switch to Features View** in the context menu.

11) Check that the title of the working area that opens is **DesignTestSys.aspx Home**.

12) Double-click **Authentication**.

13) Enable **Anonymous Authentication** and disable **Windows Authentication**.

Configuring Alfabet Expand to Update the Configuration from a Master Database via a Direct Connection

For configuring and applying new meta-model configurations to the Alfabet Application, configuration is usually not done directly in the production environment but rather on a copy of the production database in a configuration environment. Another copy of the production environment is then used to test the configuration prior to applying it to the production environment. To take over the configuration, Alfabet Expand offers a mechanism that reads data directly from the configuration database (hereafter referred to as the "master database") to the current database of the test environment (hereafter referred to as the "target database").

> (i) This method to take over configuration data is optional. Alternatively, the configuration of the master database can be stored in an AMM file, and the file can then be used to update the configuration of the target database.

The mechanism to directly take over a configuration from a master database requires Alfabet Expand to be connected to both the target and the master database. The `AlfabetMS.xml` used by Alfabet Expand to access the target database with Alfabet Expand must include the following server alias configurations:

- A server alias configuration to connect to the master database.

- A server alias configuration to connect Alfabet Expand to the target database. The server alias configuration must include settings that reference the server alias for connection of the master database.
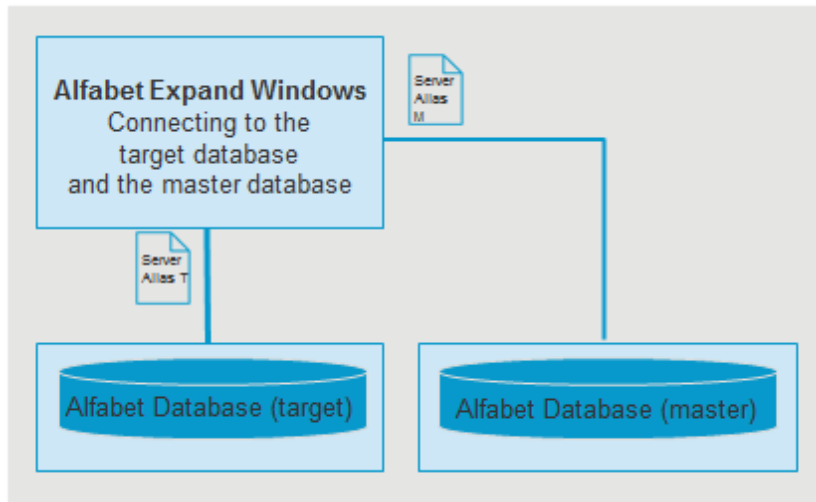
FIGURE: Alfabet components involved in the update of the meta-model from a master database

The following configuration must be added to the server alias to connect to the target database. This is done in the Alfabet Administrator:

1) In the explorer, click the **Alfabet Aliases** node.

2) In the table, select the server alias of Alfabet Expand and click the **Edit** button in the toolbar. An editor opens.

3) Go to the **Expand** tab.

4) Select the server alias name for access to the master database in the **Alias for Meta-Model Design Master Database** field.

5) Select one of the following in the **Update from Master Database Mode** field:

   • **Complete Updates**: Select this option to take over the complete configuration of the master database. The user that imports the configuration from the master database via Alfabet Expand Web cannot deselect parts of the configuration. The configuration of the target database is completely overwritten by the configuration of the master database. The option to merge the configuration is not available. For example, this option should be selected if a tester shall perform regular updates from the development environment without any knowledge about the changes that have been performed.

   • **Selective Updates**: Select this option if you would like to take over single configuration or a subset of the configurations performed on the master database. With this mechanism, the configuration of the master database can either be merged to the target configuration or can overwrite the target configuration. For example, this option should be selected if a solution designer shall update the configuration of the test environment with selected configuration changes performed for a single feature.

6) Select the **Enable Master Database Configuration** checkbox. If this option is not selected, the menu item to perform updates of the meta-model from the master database will be deactivated in Alfabet Expand.

7) If you would like to be able to additionally perform meta-model updates from AMM files via Alfabet Expand, select the **File-Based Updates Permitted** checkbox. Updates of the meta-model can be performed using the Alfabet Administrator independent of this setting.

8) Click **OK** to save your changes.

Configuring Alfabet Expand Windows to Scan the Meta-Model for Changes

Technically, multiple instances of Alfabet Expand can connect to the Alfabet database in parallel.

⚠️ When multiple instances of Alfabet Expand are used in parallel with a connection to the same Alfabet database, the following problems occur:

- Changes made to the meta-model by one user might be overwritten by changes made by another user working with another instance of Alfabet Expand and saving the same object.

- All Alfabet Expand instances are loading a working copy of the meta-model during connection to the meta-model. If the meta-model is changed by another user working with another instance of Alfabet Expand, the changes are not automatically uploaded to currently active Alfabet Expand sessions. The solution designer is not aware of changes unless using the menu option **Meta-Model** > **Reread Meta-Model**.

The following is recommended to prevent data being overwritten and inconsistencies in the Alfabet database when using Alfabet Expand:

- Do not connect with Alfabet Expand to an Alfabet database in a production environment. Configurations shall be performed in a configuration environment, stored in an AMM file and the meta-model of the production environment shall only be updated with the AMM file after testing in the configuration environment or a test environment.

- Only one instance of Alfabet Expand shall be used at a time. If it is necessary to use multiple instances in parallel, only instances of the Alfabet Expand application that connect with a server alias to the Alfabet database shall be used. The tracking of meta-model changes shall be activated in the server alias configurations of all Alfabet Expand instances.

Nevertheless, the following mechanisms are implemented to reduce the risk of overwriting concurrently made changes:

- When a user saves changes to object classes, object class properties and class keys, the connection of all other Alfabet components to the Alfabet database is closed to prevent overwriting changes made in other active Alfabet Expand instances and thus prevent inconsistencies that may result from users checking in data for an object of an object class for which the configuration has changed.

- The Alfabet Expand application can be configured to detect the changes made by other instances. If the meta-model is changed in one of the instances of the Alfabet Expand application and the changes are saved to the meta-model, the users of the other Alfabet Expand instances are informed via a bubble message on the Alfabet Expand host that the meta-model has changed. Clicking on the bubble opens a report in the center pane of Alfabet Expand that provides information about which change was made to the meta-model and the user that performed the change. This mechanism is only valid if the following applies:

  - Both instances of Alfabet Expand are an Alfabet Expand application.

  - Both applications are directly connecting to the Alfabet database with a server alias.

  - The **Track Meta-Model Changes** parameter is activated in the server alias of the Alfabet Expand instances as described below.

  - A different user name is used for login to the Alfabet Expand instances.

Instances of Alfabet Expand Web cannot track the meta-model for changes.

To activate the tracking of meta-model changes for Alfabet Expand applications:

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** ✎ button. An editor opens.

4) In the editor, go to the **Expand** tab

5) Select the check box **Track Meta-Model Changes**.

6) Optionally, you can change the amount of time to track the changes via the **Tracking Period** parameter. The default is one second.

7) Click **OK** to save your changes.

Installation and Configuration of Command Line Tools

A number of administrative tasks require the execution of command line tools. In addition, some functionalities that can be triggered or scheduled via the Alfabet user interface can alternatively be executed by command line tools.

Command line tools will be installed automatically together with the Alfabet components. They are located in the Alfabet Programs directory.

Command line tools should be executed with a Remote Alias connecting to the Alfabet Server. The command line tools can use the same Remote Alias as the Alfabet Web Application. For information about configuring a Remote Alias, see Creating a Remote Alias.

Executables for batch jobs can be started in a command line or by means of a Windows® batch job.

The documentation of the execution of batch processing tools is part of this manual and provided in general in the section About Batch Utilities for Alfabet and in the context of the documentation of the respective functionality triggered by the batch processing tools.

Special Configuration of Testing Environments

Alfabet provides some setup configurations that might be useful when working in a test environment but are not necessary or suitable for the productive environment:

- Changing the Session Timeout

- Defining a Default Login User

- Re-Routing Emails to a Defined Address for Testing

- Rendering the Alfabet User Interface in Design Mode

Changing the Session Timeout

The session time-out can be changed for the Alfabet Web Application. It is recommended t that the stand-ard time-out of 20 minutes is maintained for productive environments and that the time-out is altered only for test environments as needed.

1) Go to the Alfabet Web Application directory specified in step 12 of the basic installation.

2) Open the `web.config` file in a text editor.

3) Add the XML attribute *timeout* to the XML element *sessionState* and set it to the required time-out in minutes:

```
<sessionState mode="InProc" timeout="600"/>
```

4) Open the IIS Manager of the Internet Information Services hosting the Alfabet Web Application and change the **Idle Time-out (minutes)** of the **Advanced Settings** of the application pool used by the Alfabet Web Application to the new time-out value.

Defining a Default Login User

In a configuration or test environment, the solution designer or tester usually accesses the Alfabet user in-terface frequently with the same user login. If standard login is used, the Alfabet Web Application of the configuration or test environment can be configured to open the login shield with the user name of the test user predefined in the field for the user name:

1) Open the Alfabet Administrator.

2) Click the **Alfabet Aliases** node in the explorer. A workspace with a toolbar opens.

3) In the toolbar, click **Tools** > **Configure Alfabet.config**. An editor opens.

4) Click the Browse ▦ button on the right of the **Web Folder** field and select the main directory of the Alfabet Web Application from the directory browser. The `Alfabet.config` file in the `config` subdirectory of the selected directory opens in the editor.

5) Add a new XML element `add` to the XML element `alfaSection` and set the XML atttribute `key` to "easyin" and the XML attribute `value` to the user name that shall be used as predefined test user for login to the Alfabet user interface:

```
<alfaSection>

    <add key="easyin" value=" TestUserName "/>

    ...

</alfaSection>
```

6) Click **Save**. The changes are saved, and the editor is closed.

Re-Routing Emails to a Defined Address for Testing

When testing Alfabet functionality including sending of emails, you can configure the Alfabet Web Applica-tion to send all emails to a specified test email account, thus preventing the email from being sent to Alfabet users. This allows testing to be performed without changing the email account configuration of the Alfabet users.

To ignore email account settings of Alfabet users and to send all emails to one central email account, you can configure the server alias of the Alfabet Web Application and the Alfabet Server.

Alternatively, a test email account specification for the Alfabet Web Application can be defined in the user interface in the **Override Alfabet Settings** (`CONF_Override_Settings`) functionality and activated during testing. An activated definition in the **Override Alfabet Settings** functionality supersedes a setting in the server alias.

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer.

2) In the table, select the server alias that you want to edit and click the **Edit** button in the toolbar. An editor opens.

3) In the **Server Settings** tab, enter the name of the test email account in the **Test Receiver Email Account** field.

4) Click **OK** to save your changes.

After performing the test, delete the email address from the **Test Receiver Email Account** field.

If the **Test Receiver Email Account** field is empty, the emails are sent to the email account specified in the Alfabet user configuration for the Alfabet user that is the recipient of the email.

Rendering the Alfabet User Interface in Design Mode

The Alfabet user interface can be rendered in a design mode that is suitable for testing and configuration. The design mode is identical to the mode used when the Alfabet user interface is opened from Alfabet Expand Web or the Alfabet Expand application. In the design mode, the following additional features are added to the Alfabet user interface:

- A **Meta-Model** menu is added to the standard toolbar vvia the **Reread Meta-Model** and **Restart Web Application** functionalities. This menu is designed to take over changes to the meta-model that are made via Alfabet Expand Web to the local meta-model versions that are cached in the Web server for the Alfabet Web Application and for Alfabet Expand Web. When the design mode is used for the Alfabet user interface for testing purposes, this menu is only relevant if changes are performed with Alfabet Expand directly in the database of the testing environment. If multiple testers are working with the application simultaneously, and the **Reread Meta-Model** functionality is used in one user session, the update is valid for all currently opened sessions.

  All other user sessions are interrupted if one user performs a **Reread Meta-Model** action and re-established automatically after the action has finished. A user that sent a request during the outage might need to re-send the request.

- The technical name of the current view is displayed on the upper right of the view. Testers can communicate the technical name to the solution designer when reporting an issue to unambiguously specify the test location.

- A **Configure** button is displayed on the upper-right corner of each view. Clicking the button opens an editor that allows the user interface to be customized in the context of a specific view scheme (for example, by hiding some properties or toolbar buttons).

⚠ The design mode should never be used in a production environment.

To render the Alfabet user interface in design mode:

1) Open the Alfabet Administrator.

2) Click the **Alfabet Aliases** node in the explorer. A workspace with a toolbar opens.

3) In the toolbar, click **Tools** > **Configure Alfabet.config**. An editor opens.

4) Click the Browse 🔲 button on the right of the **Web Folder** field and select the main directory of the Alfabet Web Application from the directory browser. The `Alfabet.config` file in the `config` subdirectory of the selected directory opens in the editor.

5) Add a new XML element `add` to the XML element `alfaSection` and set the XML atttribute `key` to "appmode"and the XML attribute `value` to "Design":

```
<alfaSection>

    <add key="appmode" value="Design"/>

    ...

</alfaSection>
```

6) Click **Save**. The changes are saved, and the editor is closed.

Chapter 4:   Installation Requirements for Special Installation Scenarios

This chapter provides information about deviations or additional requirements to the installation described in the chapter Installation in some special environments:

- Using the Same Installation Parameters on Multiple Machines

- Running Alfabet in a Docker Container

  - Technical Requirements for Running Alfabet in a Docker Container

  - Preparing an Alfabet Docker Container

  - Starting the Docker Container

Using the Same Installation Parameters on Multiple Machines

If your Alfabet installation includes multiple identical deployments of Alfabet components on different hosts (for example, in the context of a fallback scenario), the installer for the Alfabet components can be run in a mode that stores all settings in a setup file or reads all settings from a setup file. The setup file has the suffix.iss. All data entered in the installer window during installation is saved, including user name and serial number.

The installer must be started via a command line under an administrative account to store parameters or to re-use stored parameters.

To store the installation parameters in a setup file, start the installer with the following command line when performing the first installation:

```
<PathToInstallationFolder>/setup.exe –r –
f1"<AbsolutePathToSaveInstallationScript>/<FileName>.iss"
```

For all other installations, the following parts of the first installation must be available on the host for the new installation:

- The installer setup file.

- The `web.config` files of the Alfabet Web Application as well as the files `Alfabet.config` and `AppSettings.config` located in the `config` subdirectory of the physical directory of the Alfabet Web Application.

  > (i) If you are using external authentication mechanisms such as federated authentication or local certificates, you must also provide all authentication-related files located in the respective sub-folders of the physical directory of the Alfabet Web Application. For more information about the authentication configurations, see Configuring User Authentication.

- The `AlfabetMS.xml` configuration file containing the server alias definitions for the Alfabet components.

To reuse the parameters stored in the setup file during the first installation:

1) Copy the `filename.iss` file from the first installation to the installation directory of the new installation.

2) Start the installer with the following command line:

```
<PathToInstallationFolder>/setup.exe -s -
f1"<AbsolutePathToSaveInstallationScript>/<FileName>.iss"
```

3) Copy the `web.config` files and, if applicable, authentication-related files of the first installation to the physical directory of the Alfabet Web Application of the new installation. The subdirectory structure must be maintained.

4) Copy the `AlfabetMS.xml` configuration file from your first installation to the installation directory of the new installation.

Running Alfabet in a Docker Container

Docker software allows applications to be run in containers that are virtualizations on an operating system level. Different to virtual machines, docker containers do not include an own operating system but use the operating system of the host.

The Alfabet application can run in an ASP.NET docker container, as distributed by Microsoft® as part of the Microsoft® Docker distribution.

> ⓘ For more information about the Microsoft® Docker distribution, see
> https://hub.docker.com/r/microsoft/aspnet/.

The following documentation covers the Alfabet-specific docker configuration. It is based on an existing implementation of the Microsoft® Docker distribution on the server host. the handling of Microsoft®-specific functionality is not part of this documentation. The technical requirements listed in the following sections provide a link to the relevant documentation provided by Microsoft.

Technical Requirements for Running Alfabet in a Docker Container

The following operating systems are supported:

- Windows Server 2016 (for creating and running the docker container)

- Windows 10 (for running the docker container)

The following installations must be available:

- Docker for Windows software must be installed and configured on the host that the docker container with Alfabet should be started on.

> ⓘ • For general information on Windows® Docker Containers, see:
>
>   • https://docs.microsoft.com/en-us/virtualization/windowscontainers/index and
>
>   • https://www.docker.com/microsoft.

- For information on how to install docker containers on Windows Server 2016, see https://github.com/docker/labs/blob/master/windows/windows-containers/Setup-Server2016.md.

- A working instance of Alfabet must be installed, configured and running.
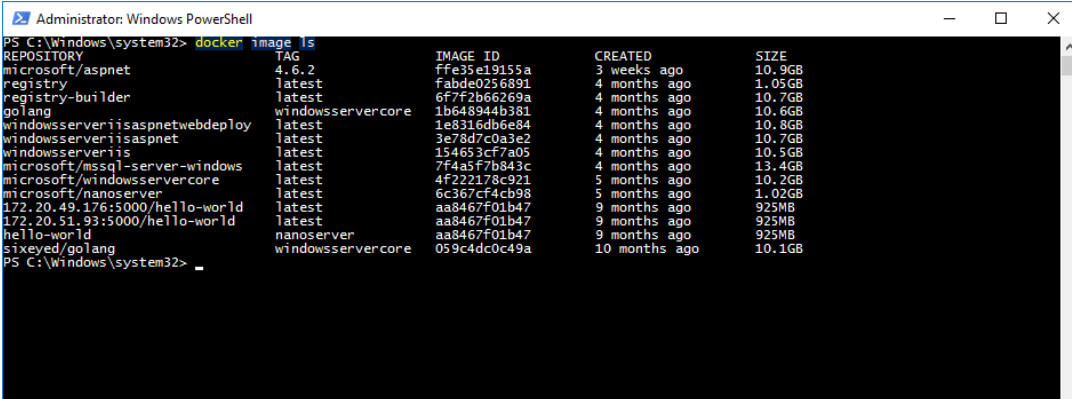
Preparing an Alfabet Docker Container

To create an Alfabet docker container on a host that meets the technical requirements:

1) On the docker enabled Windows Server 2016, create a directory for the Alfabet docker installation.

2) Copy the required Alfabet components from the installation folder of your installed Alfabet instance to this directory:

   - The **AlfabetWebApplication** folder.

   - The configured `Alfabetms.xml` file that contains the configuration of the Alfabet server alias for the Alfabet Web Application.

3) 2. Change the path to the `AlfabetMS.xml` file in the `Alfabet.config` file in the `config` subdirectory of the `AlfaWebApplication` directory to point to `c:\inetpub\wwwroot\Alfabetms.xml`.

4) Open Windows PowerShell as administrator.

5) The docker image `microsoft/aspnet:4.6.2` (or a higher release number) should be preset in your docker repository. To ensure that it is available, execute the following command:

   ```
   docker image ls
   ```

   You should see `microsoft/aspnet` with the tag `4.6.2` (or later) in the results, as shown below:

   If the results do not contain `microsoft/aspnet` with the tag `4.6.2`, execute the following command:

   ```
   docker pull microsoft/aspnet:latest
   ```

After the process has finished, make sure that the windows/aspnet image is now available by executing again the command:

```
docker image ls
```

6) Navigate to the folder containing your files, using the following command:

```
cd <path to your folder>
```

7) Validate that the folder contains a `AlfabetWebApplication` subdirectory and the `Alfabetms.xml` file.

Starting the Docker Container

After the ASP.NET image has been successfully downloaded and the `Alfabet.config` file has been reconfigured as described in the section Preparing an Alfabet Docker Container, start the container from the Windows PowerShell:

1) Navigate to the folder containing your files, using the following command:

```
cd <path to your folder>
```

2) Start the container with the command:

```
docker run-p 80:80-v="[path to Alfabet Web
Application]c:\inetpub\wwwroot"--name Alfabet mi-crosoft\aspnet:4.6.2
```


3) To determine which IP address is used for the started Alfabet docker instance, execute the command:

```
docker inspect-f
'{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' $(docker ps-q-n
1))
```

> ℹ️ If you restart the container, the IP address will change.
>
> You can use the following command to start the container with a static IP address:
>
> ```
> docker run-d-p 80:80--ip=172.26.148.155-v="[path to Alfabet
> Web Application]c:\inetpub\wwwroot"--name Alfabet
> microsoft\aspnet:4.6.2
> ```
>
> In order to assign a proper docker IP, you need to understand docker container networking. Information is available at: https://docs.docker.com/engine/userguide/networking/

The following instructions are useful for providing access to the Alfabet docker instance:

- For the configuration of HTTPS connections to the Alfabet docker instance, the following blog provides information about the creation of self-issued certificates:

  https://blogs.msdn.microsoft.com/zxue/2016/12/19/adding-https-support-to-individual-windows-containers-using-self-issued-certificates/

- for information about configuring a reverse proxy server to provide external access to the Alfabet docker instance, read the following:

  http://jasonwilder.com/blog/2014/03/25/automated-nginx-reverse-proxy-for-docker/

Chapter 5:   Considering Security Issues

This chapter informs about security mechanisms implemented in Alfabet as well as giving an overview of basic requirements for a secure implementation environment.

The following information is available:

- Best Practice Security Implementation

- Secure Data Exchange Between the Alfabet components

  - Security Mechanisms for Data Transfer Between Components

  - Browser Session Management

    - Session Timeout

    - Client Browser Verification

    - User Verification

  - Removing ASP.NET Version Information from the HTTP Header

  - Session Cookies of the Alfabet Web Application

  - Hiding Web Server Content from Potential Attackers

  - Directories Used by the Alfabet Components Impacted by Antivirus Software

  - Activating Anti-Virus Check for Upload of Documents

  - Executing Configured Reports with a Different Database User

    - Using an Alfabet Internal ReadOnly User for Report Execution

    - Configuring Database Users with Explicit Access Rights for Individual Configured Reports

- Configuring User Authentication

  - Understanding User Authentication

    - Windows Sign-On

    - Authentication via a Company Authentication Portal

    - Authentication via Federated Authentication

    - Authentication via Client Authentication Certificates

    - Standard User Login

    - Customer-Specific Authentication Mechanisms

  - Configuring Windows Sign-On

    - Activating Windows Sign-On

    - Configuring the Alfabet Web Application to Accept Windows Sign-On

    - Configuring a Named User

Best Practice Security Implementation

A number of mechanisms for secure data transfer and access control to the Alfabet components are implemented. You can choose between the available access permission concepts according to your company's infrastructure requirements and security concepts.

In addition, the environment for implementation of the Alfabet components should be secured according to the security standards of your organization.

It is recommended to take the following measures for secure data transmission, processing, and storage:

The following security and access permission concepts are implemented for connections between the Alfabet components:
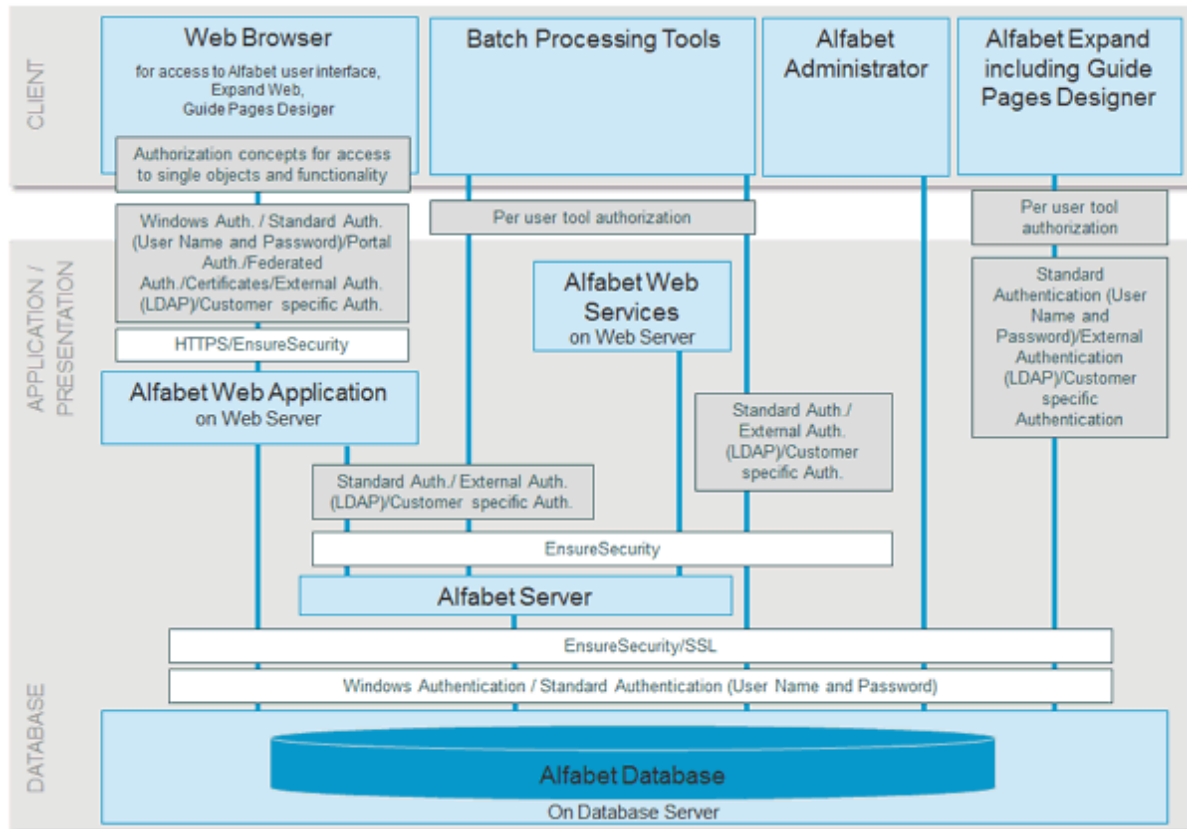
FIGURE: Security mechanisms for access to Alfabet

The figure above distinguishes between security mechanisms implemented for securing access from one component to another (displayed in white fields) and authentication and authorization mechanisms that are implemented for authentication and authorization of individual Alfabet users (displayed in grey fields):

- The implementation of secure data transfer between components including browser session management is described in the section Secure Data Exchange Between the Alfabet components . It includes data encryption on the network level that is not specific to Alfabet but part of the security policies of your company.

- The implementation of user authentication is described in the section Configuring User Authentication. It includes all mechanisms that ensure that only authorized users can access Alfabet. For each component of Alfabet access permission can be configured individually. For example, user authentication via a Web portal can be used for access of users to the Alfabet user interface while authentication via user name and password is used for access of system designers to Alfabet Expand

- The implementation of user authorization is performed with the configuration tool Alfabet Expand and is described in the reference manual *Configuring Alfabet with Alfabet Expand*. Authorization performed via an external data source is described separately in this manual in the section Configuring User Authorization Based on Windows Sign-On Data or Data from Federated Authentication.

- User tool authorization is described for each relevant tool in the section Enabling a User to Access Tools for Configuration and Administration. The authorization to use Alfabet Expand and/or batch processing tools is defined separately for each Alfabet user.

Secure Data Exchange Between the Alfabet components

The following security issues must be taken into consideration when making Alfabet available to the user community:

- Security Mechanisms for Data Transfer Between Components

- Browser Session Management

  - Session Timeout

  - Client Browser Verification

  - User Verification

- Removing ASP.NET Version Information from the HTTP Header

- Session Cookies of the Alfabet Web Application

- Hiding Web Server Content from Potential Attackers

- Directories Used by the Alfabet Components Impacted by Antivirus Software

- Activating Anti-Virus Check for Upload of Documents

- Executing Configured Reports with a Different Database User

  - Using an Alfabet Internal ReadOnly User for Report Execution

  - Configuring Database Users with Explicit Access Rights for Individual Configured Reports

Security Mechanisms for Data Transfer Between Components

For secure data transfer and access control between components, the encryption and authentication mechanisms of the network infrastructure apply (for example, IPSec, FIPS).

Independent of the encryption mechanisms implemented for data transfer, user passwords are always encrypted during storage (non-symmetric SHA-1 encryption) and transmission (symmetric encryption using the Rijndael algorithm). Both encryption mechanisms for user passwords are compliant with FIPS.

The following security mechanisms require a configuration of the Alfabet components. The configuration required to implement these mechanisms in the Alfabet components is described as part of the installation procedure in the chapter Installation. The required configuration of the local network and servers is not part of this documentation.

Database Access Permissions

Each Alfabet component that requires access to the Alfabet database must log in to the database server to access the Alfabet database. Login is usually performed via user name and password. For Microsoft® SQL Server® Windows authentication is also available. For the execution of configured reports, a restricted level of database access can be added by configuring the Alfabet Web Application to log in to the database server with a different database user with restricted access permissions. The required configuration is described below in the section Executing Configured Reports with a Different Database User.

.NET Ensure Security

.NET EnsureSecurity can be used to secure the communication channels between the Alfabet components. The machines on which the Alfabet components are installed must be members of the same active directory to use.NET EnsureSecurity.

HTTPS

A secure channel can be implemented between the Alfabet Web Application and the client's web browser using the HTTPS protocol. This must be configured in the Internet Information Services® Manager. The necessary server and (if desired) client certificates must be created using an existing public key infrastructure (PKI).

SSL

If the Alfabet database is installed on a Microsoft® SQL Server®, Secure Sockets Layer (SSL) can be used for connections between the Alfabet components and the Alfabet database.

You can force the Alfabet Web Application to accept only SSL-secured connections. In this case, all HTTP requests will be redirected to HTTPS. Please note that this option should only be implemented if HTTPS is implemented on all client hosts. The required code to implement strict usage of SSL is already available in the example `web.config` file delivered with the release, but deactivated as comment. Activate the following lines of code in the `system.webServer` XML element of the `web.config` file to enforce SSL:

```
<rewrite>

    <rules>

        <rule name="HTTP to HTTPS redirect" stopProcessing="true">

            <match url="(.*)" />

            <conditions>

                <add input="{HTTPS}" pattern="off" ignoreCase="true" />

            </conditions>

            <action type="Redirect" url="https://{HTTP_HOST}/{R:1}"
            redirectType="Permanent" />

        </rule>

    </rules>

    <outboundRules>

        <rule name="Add Strict-Transport-Security when HTTPS" enabled="true">

            <match serverVariable="RESPONSE_Strict_Transport_Security"
            pattern=".*" />

            <conditions>

                <add input="{HTTPS}" pattern="on" ignoreCase="true" />

            </conditions>

            <action type="Rewrite" value="max-age=31536000; includeSubDomains;
            preload" />

        </rule>

    </outboundRules>

</rewrite>
```

Expect CT Security Headers

The Alfabet Web Application optionally supports the Expect CT Security Headers feature on HTTPS connections if the browser used to open the Alfabet user interface supports the feature. The following settings must be added to the `Alfabet.config` file of the Alfabet Web Application to enable the support.

```
<add key="expect_ct_enable" value="true"/>

<add key="expect_ct_max_age" value="0"/>

<add key="expect_ct_report_url" value="https://fqdn/report-ct"/>

<add key="expect_ct_enforce_policy" value="false"/>
```

> (i) For information about the location and editing options of the `Alfabet.config` file, see Configuring the web.config Files for the Alfabet Web Application.

The following values must be set for the XML attribute `value` in combination with the listed values of the XML attribute `key`:

key attribute	value attribute
`expect_ct_enable`	Enter `true` to enable Expect CT Security Header support. Enter false to disable the feature and ignore all other settings for Expect CT Security Headers in the `Alfabet.config` file.
`expect_ct_max_age`	Define the number of seconds after reception of the Expect CT Security Header field during which the user agent should regard the host of the received message as a known host. The default value is zero.
`expect_ct_report_url`	Optionally, define a URL to send problem reports to.
`expect_ct_enforce_policy`	If set to true, any policy violation will lead to a block of the browser access to the Web server for the time specified with the `expect_ct_max_age` key.

Browser Session Management

A user session with a session ID is opened when the user accesses the URL of the Alfabet Web Application. The session ID will remain the same after login of the user to the Alfabet user interface. On login, a sub-session ID is created that is valid for the current user session in the current browser tab. If a user opens a link on the Alfabet user interface in a new tab or browser window, a new sub-session ID is created for the new tab or browser window but the session ID remains the same. Session integrity is checked on each request to the Alfabet Web Application.

- The information about the user and the session ID is integrated into the sub-session ID to check validity of user requests to the Alfabet Web Application. If the sub-session ID of user request is not corresponding to the current user and session ID, the sub-session ID is set to NULL, the session is terminated and the login screen is displayed.

- The sub-session ID of a browser tab remains the same during the whole user session. If the sub-session ID of a request from a browser tab does not match the one of the previous requests, the session is terminated and the login screen is displayed.

The session terminates when the user logs out of the Alfabet user interface. Logout is applied to all tabs and browser windows opened with the same session ID.
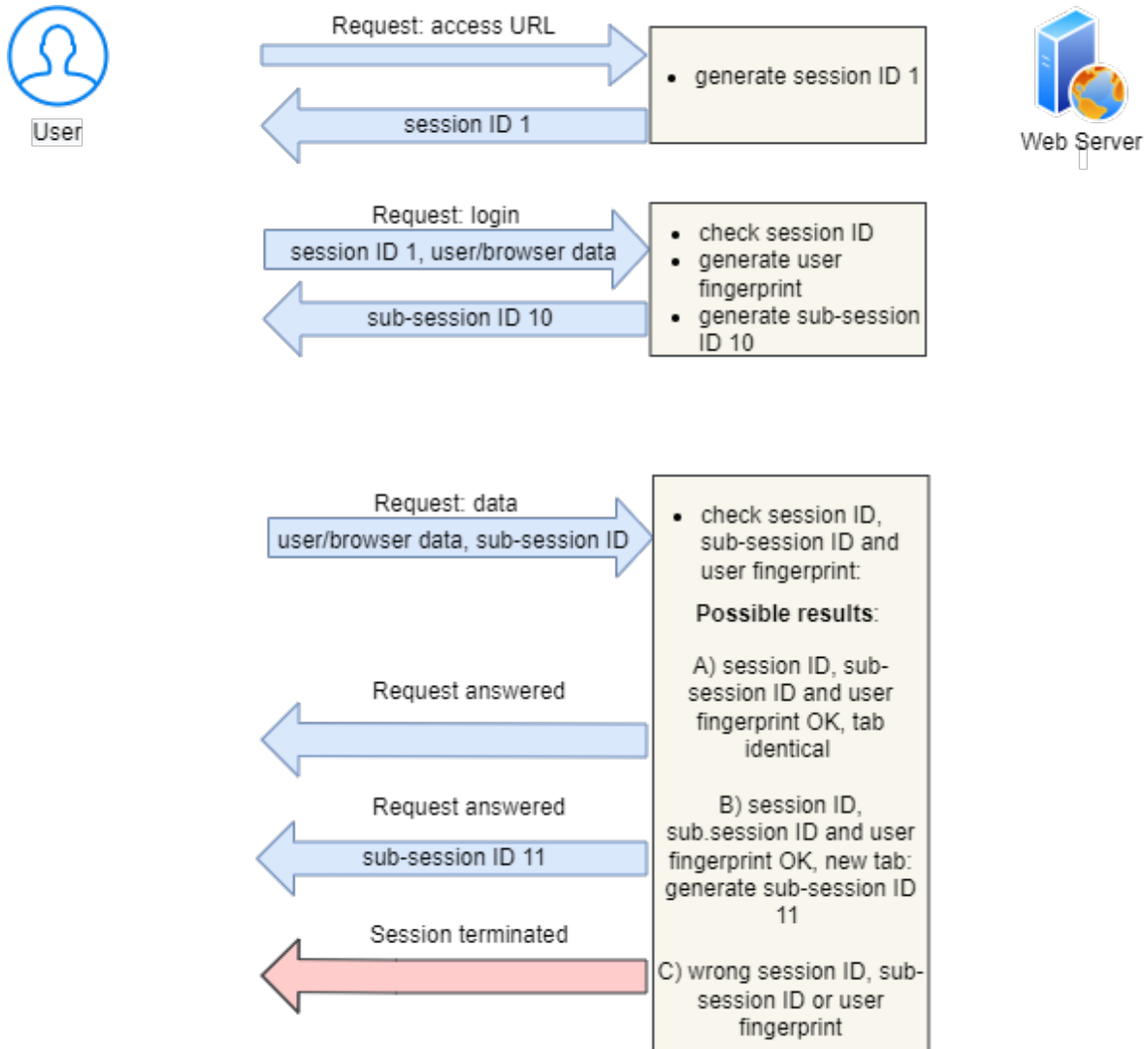
FIGURE: Security Checks in Session Management

A number of additional mechanisms have been implemented that check validity of a user session:

- Session Timeout

- Client Browser Verification

- User Verification

Session Timeout

Inactive sessions will time out after 20 minutes. After timeout, a new screen is displayed to the user informing him/her that the session expired and that he/she must re-login to continue working with Alfabet. After re-login, the page the user was currently working with is displayed again.

The session time out can be changed for the Alfabet Web Application. It is recommended that the standard time out of 20 minutes is maintained for productive environments.

To change the allowed inactivity time for user sessions:

1) Go to the Alfabet Web Application directory in the installation directory of the Alfabet components.

2) Open the `web.config` file in a text editor.

3) Add the XML attribute *timeout* to the XML element *sessionState* and set it to the required timeout in minutes:

```
<sessionState mode="InProc" timeout="600"/>
```

4) Open the IIS Manager of the Internet Information Services hosting the Alfabet Web Application and change the **Idle Time-out (minutes)** of the **Advanced Settings** of the application pool used by the Alfabet Web Application to the new timeout value.

Client Browser Verification

A security mechanism has been implemented that ensures that all requests sent to the Alfabet Web Application are sent from the same client browser. If the client host check is not positive, the session will be terminated and the login screen displayed for re-login. This security check takes network parameters into account. For example, changing from LAN to WLAN during a session will terminate the session.

The security check for IP context validation can be deactivated in the `Alfabet.config` file of the Alfabet Web Application. You should only deactivate the check if your enterprise encounters the frequent termination of active sessions:

1) Open the Alfabet Administrator.

2) Click the **Alfabet Aliases** node in the explorer. A workspace with a toolbar opens.

3) In the toolbar, click **Tools** > **Configure Alfabet.config**. An editor opens.

4) Click the Browse ⬚ button on the right of the **Web Folder** field and select the main directory of the Alfabet Web Application from the directory browser. The `Alfabet.config` file in the `config` subdirectory of the selected directory opens in the editor.

5) Add the following XML element as a child XML element to the XML element `AlfaSection`:

```
<add key="ValidateSessionContext" value="false"/>
```

> ⓘ The example `Alfabet.config` files delivered with the installation already contain the code but with the XML attribute `value` set to `"true"`.

User Verification

Information about the user currently logged in and the user profile the user is currently logged in with is encoded into the URL of all Alfabet views opened during a user session. This will effectively prevent a URL being copied and sent to a different user unintentionally as the receiving user will not be able to open this link.

Please note that if a URL must be sent, the **Express View** capability must be used instead. For more information, see the section [Configuring the Express View (Email) Capability](#) and the section *Sending and Receiving Express Views* in the reference manual *Getting Started with Alfabet*.

Removing ASP.NET Version Information from the HTTP Header

The ASP.NET version information is part of the HTTP header. To enhance security, this information can be removed from the HTTP header by setting the XML attribute `enableVersionHeader` of the XML element `httpRunTime` in the `web.config` file to `false`:

```
<httpRunTime enableVersionHeader="false">
```

Session Cookies of the Alfabet Web Application

The Alfabet Web Application is based on ASP.NET. By default, ASP.NET uses cookies to identify which requests belong to a particular session. The default settings of the Alfabet Web Application ensure the correct handling of cookies. If your company's security policies include special requirements and restrictions for cookies, or you are using multiple installations of the Alfabet components, you can configure the handling of cookies in the `web.config` file of the Alfabet Web Application:

⚠️ It is recommended that the Web server is setup to enforce the use of HTTPS for each request. You can additionally configure the Alfabet Web Application to enforce acceptance of cookies on SSL secure connections only.

- The cookie name configured in the session state settings must be unique for each application running on the server. Please note that the value of the XML attribute `cookieName` in the XML element `sessionState` in the `web.config` file must be specified differently for the `web.config` file used in the test environment and for the `web.config` file used in the production environment file. For example:

  ```
  <sessionState cookieName="Alfabet912" mode="InProc" />
  ```

- Disabling cookies: If cookies are not available, a session can be tracked by adding a session identifier to the URL. To disable cookies, set the XML attribute `cookieless` of the XML element `sessionState` in the `web.config` file to "true". The XML element `sessionState` is a child element of the XML element `system.web` and is included in the example `web.config` file in the `Example` subdirectory of the installation directory of the Alfabet Web Application.

- Forcing the cookies to be SSL secure: By default, the cookies sent by the Alfabet Web Application are sent without requiring SSL security. That means that cookies can be sent and accepted by both HTTPS and HTTP connections. If you want HTTPS to be used for the connection between the Alfabet Web Application and the browser and want to ensure that the session cookies of the Alfabet Web Application are only valid for HTTPS connections, you can add a child XML element `httpCookies` to the XML element `system.web` of the `web.config` file with a parameter

`requireSSL` set to `"true"`. Users that try to access the Alfabet Web Application on HTTP connections cannot access the Alfabet user interface when this setting is specified and will see the error message "Session data unavailable".

```
<configuration>

    ...

    <system.web>

        <httpCookies httpOnlyCookies="false"
        requireSSL="true"/>

        ...

    </system.web>

</configuration>
```

Hiding Web Server Content from Potential Attackers

The `web.config` file of the Alfabet Web Application can be configured to hide information available on the web server from potential attackers. For example, the following content can be hidden to users for security reasons:

- The OPTIONS method of the Internet Information Services® for the Alfabet Web Application provides a list of allowed methods on the Web server. It is recommended that the OPTIONS method is disabled in order to hide the information from potential attackers.

- The TRACE method provides a mean to read the information contained in the HTTP request sent to the Web server. This includes cookies and Web authentication strings and it is therefore recommended that the TRACE method is disabled in order to hide the information from potential attackers.

- Single characters or sequences of characters can be defined that will lead to a denial of URL processing if the specified character(s) are included in the URL. For example, excluding URLs that contain.. will prevent attackers from accessing the content of the next higher directory of an URL.

To disable the OPTIONS method, in the `web.config` file of the Alfabet Web Application. The code is displayed below with the surrounding parent elements:

1) Go to the Alfabet Web Application directory in the installation directory of the Alfabet components.

2) Open the `web.config` file in a text editor.

3) Add the following code to the XML element `requestFiltering` that is a child of the XML element `security` located in the `XML element system.webServer`:

```
<system.webServer>

    <security>

        <requestFiltering>

            <verbs>

            <add verb="OPTIONS" allowed="false" />

            <add verb="TRACE" allowed="false" />

            </verbs>
```

```
            <denyUrlSequences>

                <add sequence=".." />

                <add sequence=":" />

            </denyUrlSequences>

            <hiddenSegments>

                <add segment="SAML" />

            </hiddenSegments>

        </requestFiltering>

    </security>

</system.webServer>
```

Directories Used by the Alfabet Components Impacted by Antivirus Software

The performance and stability of the Alfabet application may be affected by anti-virus software. Therefore, it is recommended that you deactivate the **On Access Scan** option for all directories that the Alfabet components use to read or write data for virus scanners installed on Windows® servers hosting Alfabet components. If this is not done, the performance of the access to Alfabet may be decreased by a factor of 10.

Alfabet components write data to the following directories:

Component	Directories Used on Component Host
Web Server hosting the Alfabet Web Application	`Alfabetwebapplication\runtime`(`Alfabetwebapplication` directory location is specified during the installation process.)
Alfabet Server	<User temp directory of the service account>`\Alfabet\`<serveralias>

Alfabet components read data from all directories specified during the installation process.

Recommendations for the settings concerning the database server are not included in the overview because they are the responsibility of the database manager and are independent of the applications accessing the database.

Activating Anti-Virus Check for Upload of Documents

An anti-virus check for documents is implemented and activated by default. The scan will be performed with the anti-virus scanner implemented on the server hosting the Alfabet Web Application. If no anti-virus scanner is explicitly implemented, the anti-virus capabilities of the underlying Windows® operating system will be used to scan documents during upload.

The options for anti-virus scans can be deactivated, but it is recommended to leave them activated.

The following settings are available for activation or deactivation of anti-virus scanns in the Alfabet.config file and the server alias configuration of the Alfabet Web Application:

- Anti-virus check of documents on upload to the **Internal Document Selector** is activated with the `scan_malware` key in the `Alfabet.config` file:

  ```
  <add key="scan_malware" value="true"/>
  ```

- Anti-virus check of Microsoft® Word® and PowerPointMicrosoft® PowerPoint® templates prior to each publication process via the **Publications** functionality is activated with the `scan_templates_for_malware_when_publishing` key in the `Alfabet.config` file:

  ```
  <add key="scan_templates_for_malware_when_publishing" value="true"/>
  ```

- Anti-virus check of Microsoft® Word® and PowerPointMicrosoft® PowerPoint® templates prior to uploading the template into the publication definition in the **Publications** explorer in Alfabet Expand is activated with the option **Scan uploaded files for malware** in the **Expand** tab of the server alias configuration of the Alfabet Web Application.

> ⓘ For information about changing the settings in the `Alfabet.config` file, see Configuring the web.config Files for the Alfabet Web Application.
>
> For information about changing the server alias configuration, see Working with Alfabet Aliases.

Executing Configured Reports with a Different Database User

The database user that is used for connections from the Alfabet Web Application to the database server in order to access the Alfabet database has very extensive rights. To enhance security, you can configure all or individual reports to be executed with a database user that is different than the standard database user for connections from the Alfabet Web Application. There are two levels of restrictions:

- The Alfabet Web Application can be configured to use a user with restricted access permissions to execute all configured reports. The database user is automatically created on the database server during installation and has ReadOnly access permissions to the database. This option should not be selected if data cube-based configured reports shall be executed. The access permissions of the automatically created database user with ReadOnly access do not allow cube processing.

- Database users with specific access permissions can be defined by the customer on a Microsoft® SQL Server® hosting the Alfabet database. The configured reports can then be configured to be executed with one of the customer defined database users. If a user opens a configured report and the database user defined to execute the configured report does not allow report execution, an error message will be displayed informing the user about the inadequate access permissions for report execution and asking him/her to contact the system administrator for resolution of the issue. This method is currently only available for configured reports of the type `Query` and `NativeSql`. For configured reports of the type `Query` and `NativeSql`, the configured database users supersedes the configuration to use the Alfabet internal ReadOnly user.

The following sections describe

all required configuration steps for the implementation of each method:

- Using an Alfabet Internal ReadOnly User for Report Execution

- Configuring Database Users with Explicit Access Rights for Individual Configured Reports

Using an Alfabet Internal ReadOnly User for Report Execution

During installation of the Alfabet components, a ReadOnly database user is automatically created on the database server for the Alfabet database. This ReadOnly user is re-created after each meta-model update. The Alfabet Web Application can be configured to connect to the Alfabet database with the database user with ReadOnly access rights when executing configured reports.

> ⚠ Please note that cube-based reports cannot be executed when the database connection is set up with the ReadOnly database user.

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** 🖉 button. An editor opens.

4) In the editor, go to the **Server Settings** tab and open the **General** tab.

5) Select the **Use ReadOnly User for Report Execution** checkbox.

6) Click **OK** to save your changes.

Configuring Database Users with Explicit Access Rights for Individual Configured Reports

Customer-defined database users that are available on the database server can be used to connect the Alfabet Web Application to the Alfabet database when executing configured reports of the type `Query` or `NativeSQL`.

One of the database users can be used as default user for all configured reports of the type `Query` or `NativeSQL`. Additional database users can be specified for execution of one or multiple individual configured reports.

> ⚠ The **Use ReadOnly User for Report Execution** setting in the server alias of the Alfabet Web Application is ignored if a definition of customer-defined database users for report execution is available. The customer-defined database users are only applied to configured reports of the type `Query` or `NativeSQL`. All other configured reports are then executed for the database user with extended rights (the standard database user the Alfabet Web Application connects with to the database server).

The configuration includes three steps:

First step: Definition of database users on the database server host

The database users must be created and the access permissions configured on the database server. This is typically done by a database administrator. For details see the documentation of the database server used.

> ⚠ The access permissions to the Alfabet database for database users on the database server might be overwritten when the meta-model is updated (for example, when migrating to another release or taking over a configuration to the production environment via an AMM file). It is recommended that you auto-run ADIFschemes for re-constitution of the database user rights after migration. For more information about ADIF schemes, see the reference manual *Alfabet Data Integration Framework*.

Second Step: Configuring the Alfabet Web Application to find the database users

The login credentials of all database users that shall be used to execute configured reports must be defined in the configuration of the meta-model of the current database in Alfabet Expand.

> ⓘ For information about accessing and working with the configuration tool Alfabet Expand, see the reference manual *Configuring Alfabet with Alfabet Expand*.

1) Open Alfabet Expand and go to the **Presentation** tab.

2) Expand the **XML Objects** > **Administration** node.

3) Right-click *DatabaseUsers* and select **Edit XML** in the context menu. The XML object opens in the center pane.

4) In the XML object, add one or multiple XML elements `DatabaseUser` to the XML root element `DatabaseUsers` and set the following XML attributes for each XML element `DatabaseUser`:

   - `Name`: Define a unique name for the database user. The name will be displayed in the drop-down field in the **Database Restricted User** attribute of configured reports of the type `Query` or `NativeSQL` that allows a database user to be assigned to the configured report.

   - `DBUserName`: Enter the database user name that shall be used for authentication against the database server. Server variables can be used to define the database user name. For more information on server variables, see <u>Defining Connections Based On Server Variables</u>.

   - `DBUserPassword`: Enter the database user password that shall be used for authentication against the database server. Server variables can be used to define the database user password. For more information on server variables, see <u>Defining Connections Based On Server Variables</u>.

   - `IsDefaultForReports`: Specify "`true`" if the database user shall be used for execution of all configured reports of the type `Query` or `NativeSQL` that do not have a database user explicitly defined. If the XML attribute is specified "`false`"or is not set, the database user will not be the default user.

   > ⚠ Only one of the defined database users can be defined as the default for reports.

5) In the toolbar, click the **Save** 💾 button to save your changes.

Step 3: Configuring reports to be executed with one of the defined database users

If the database user specification in the XML object *DatabaseUsers* includes the definition of a default user, all configured reports of the type `Query` or `NativeSql` are executed with the default database user without any further configuration required on the level of report configuration. If no database user is defined as default or if a default is defined, but a report shall be executed with another database user than the default database user, the database user for execution must be explicitly defined in the configured report's attributes.

> ⓘ For information about accessing and working with the configuration tool Alfabet Expand, see the reference manual *Configuring Alfabet with Alfabet Expand*.

1) Open Alfabet Expand and go to the **Reports** tab.

2) In the explorer, click the node of the report for which you want to define a database user.

3) In the attribute window, select a database user in the drop-down field in the **Database Restricted User** field.

Configuring User Authentication

For each user that shall access Alfabet, data required to authenticate the user is stored in the Alfabet database. The user data is maintained by a system administrator in the Alfabet user interface or via the Alfabet Administrator. The data stored about a user in the Alfabet database specifies a user name for login as well as details about the access permissions that are granted to this individual user:

- The type of the user can only be `NamedUser` or `Anonymous`. Named users have full read/write access to Alfabet. Anonymous users only have read only access permissions to the Alfabet user interface. Access to Alfabet components for configuration and administration of Alfabet is limited to named users.

- Users are associated with one or multiple user profiles. User profiles are customer configured in the configuration tool Alfabet Expand. They define the scope of functionality that is available to the user by providing the navigation to the relevant functionality. User profiles are also coupled with read/write or read only access permissions that supersede the access permissions granted by the user type. Therefore, a named user that logs in to Alfabet with a read only user profile cannot edit any objects. Each named user must have one user profile assigned. Multiple user profiles can be assigned to a single named user. The user can then switch between the user profiles while working with Alfabet.

  Anonymous users do not have a user profile assigned. Instead, a read only user profiles is configured to be used for anonymous access. All anonymous users access Alfabet with that user profile. Anonymous users cannot switch between multiple user profiles.

- A user can only access the configuration tool Alfabet Expand or user one of the batch processing utilities that are part of the Alfabet components if access is specifically granted via the user data.

The individual access permissions for the user are checked against the user data in the Alfabet database when the user accesses Alfabet via one of the following components:

- Alfabet user interface accessed via a Web browser

- Alfabet Expand

- Guide Pages Designer

- batch processing utilities

The Alfabet authentication mechanism serves a large, heterogeneous, and distributed user community. Therefore, there are two distinctly different authentication setups:

- **Single Sign-On**: With enterprise authentication, the user can automatically sign-on to Alfabet without providing a password when he/she is already authenticated by either his/her Windows user name, by login to a company authentication portal, via a client side Client authentication certificate, or via the federated authentication mechanisms that are implemented in the company.

- **Standard login**: The user authentication is performed within Alfabet. Upon login, the user is prompted to provide a user name and password.

> ⓘ The user's login credentials can be checked against the login data from an external data source. The configuration for this setup is described separately in the section Configuring the Alfabet Web Application to Perform User Authentication Based On User Data from an External LDAP Data Source of chapter Integrating Data from External Sources.

- **Customer defined login**: Proprietary additional authentication requirements can be considered after or instead of the supported standard single sign-on mechanisms or standard user login.

The use of some of the implemented authentication mechanisms is restricted to a subset of the Alfabet components:

Authentication Mechanism	Description	Available For
Standard Authentication	The user logs in with a user name and password.	• Alfabet user interface, Alfabet Expand Web and the Guide Pages Designer accessed via a Web browser • Alfabet Expand application • batch processing utilities • Alfabet RESTful API
Standard Authentication against an external data source	User authentication is done via an external database such as an LDAP server.	• Alfabet user interface, Alfabet Expand Web and the Guide Pages Designer accessed via a Web browser • Alfabet Expand application • batch processing utilities
Customer specific authentication mechanism	User authentication is done via a mechanism implemented for an individual customer via an assembly.	• Alfabet user interface, Alfabet Expand Web and the Guide Pages Designer accessed via a Web browser • Alfabet Expand application • batch processing utilities • Alfabet
Windows Authentication	The login credentials for Windows® are also used to log in to Alfabet.	• Alfabet user interface accessed via a Web browser • Alfabet Expand Web

Authentication Mechanism	Description	Available For
Portal Authentication	Alfabet is provided via a portal and the login credentials for the portal are also used to log in to Alfabet.	• Alfabet user interface accessed via a Web browser • Alfabet Expand Web
Federated Authentication	Alfabet can be part of a single sign-on (SSO) system that allows a single user authentication process across multiple IT systems or even organizations.	• Alfabet user interface accessed via a Web browser • Alfabet Expand Web
Certificates	Authorization certificates within a public key infrastructure can be used to authenticate users for login to Alfabet.	• Alfabet user interface accessed via a Web browser • Alfabet Expand Web

The authentication mode must be defined in the server alias configuration used for database access. Each can only use one of the authentication mechanisms, but users can access the same Alfabet database via different components with different authentication mechanisms. For example, if Windows authentication shall be used for some users while others shall log in with user name and password, two Alfabet Web Application s with different authentication mechanisms must be installed in parallel, both accessing the same Alfabet database.

For component that connect to the Alfabet Server, authentication is set up in the remote alias configuration of the client application and the authentication configuration of the server alias is ignored. With this setting, only one Alfabet Server installation is required to run client applications that are using different authentication mechanisms.

The client application can use a different AlfabetMS.xml configuration file than the Alfabet Server, containing only the remote alias configuration required by the specific client application. The Alfabet Server can be configured to control the authentication settings of the remote alias configurations. When the Alfabet Server is configured to control authentication, the remote alias configurations of the relevant clients must be available both in the `AlfabetMS.xml` file of the Alfabet Server and of the clients. When a client connects to the Alfabet Server, the name of the remote alias configuration is communicated to the Alfabet Server and the Alfabet Server ignores the authentication method communicated by the remote client but reads the authentication configuration from the remote alias configuration of its own `AlfabetMS.xml` configuration file.

Understanding User Authentication

In this section, the mechanisms of authentication are described in detail for the authentication mechanisms supported by Alfabet:

• Windows Sign-On

• Authentication via a Company Authentication Portal

- [Authentication via Federated Authentication](#)

- [Authentication via Client Authentication Certificates](#)

- [Standard User Login](#)

- [Customer-Specific Authentication Mechanisms](#)

Windows Sign-On

When a user opens the Alfabet user interface via a Web browser, either the current Windows user credentials are automatically sent to the Internet Information Services® or the user is provided with a login screen for authentication against the Internet Information Services (with a local or a trusted Windows domain account). The Alfabet Web Application impersonates the authenticated user and verifies whether a named user with a user name matching the user's Windows user name already exists in the Alfabet database.

If the user name is found, the user profiles attached to the user will then be displayed and can be selected by the user.

If the corresponding user cannot be found in the Alfabet database, the Alfabet Web Application can be configured either to deny access or to create a Alfabet user of the type Anonymous in the database. For security reasons, a random password is assigned to the user to disable re-login via standard login. If the Alfabet Web Application allows anonymous access, the ReadOnly user profile configured for anonymous access will be displayed. The administrator can later change the **Type** attribute of the newly created anonymous user to `NamedUser` and assign user profiles to the user.

> ⓘ Anonymous access is especially relevant for the configuration of access to Alfabet views from external applications or for links in emails sent via Alfabet.
>
> For Alfabet Expand Web, anonymous access is not suitable because access to configuration functionalities should be limited to solution administrators who should have additional access permissions defined. Therefore, Alfabet Expand Web cannot be accessed by anonymous users. Furthermore, new users cannot be added to the Alfabet database as anonymous users on their first access. For information about configuring user profiles for anonymous access, see *Defining a User Profile for Anonymous Users* in the reference manual *User and Solution Administration*.
>
> For information about configuring the Alfabet Server to allow anonymous access, see either
>
> - [Access Permissions To Open the Alfabet User Interface](#) in the section [Activating the Dispatch of Email Notifications in Alfabet](#) or
>
> - [Managing Access Permissions for Access from External Applications](#) in the section [Accessing the Alfabet Database with External Applications](#).

A login request from a user account that does not provide domain and user name information will be routed to a guest login. (If, for example, the user chooses to disable sending domain and user information.)

The authentication via Windows sign-on can only be performed using the Alfabet Web Application. All other Alfabet components must be configured to use standard login.

Authentication via a Company Authentication Portal

If Alfabet is integrated into a company web authentication portal, the Alfabet Web Application can be configured to read the user name of the user logged in to the portal from the HTTP server variable created from a customer defined HTTP header that is sent with the request for opening the web client. The HTTP header must contain the user name of the user logged in to the portal. The Alfabet Web Application verifies whether a user of the type `NamedUser` with a matching user name already exists in the Alfabet database.

If the user name is found, the user profiles attached to the user will then be displayed and can be selected by the user. If the corresponding user cannot be found in the Alfabet database, the Alfabet Web Application can be configured either to deny access or to create a Alfabet user of the type `Anonymous` in the database. If the Alfabet Web Application allows anonymous access, the ReadOnly user profile configured for anonymous access will be displayed. The administrator can later change the **Type** attribute of the newly created anonymous user to `NamedUser` and assign user profiles to the user.

> ⓘ Anonymous access is especially relevant for the configuration of access to Alfabet views from external applications or for links in emails sent via Alfabet.
>
> For Alfabet Expand Web, anonymous access is not suitable because access to configuration functionalities should be limited to solution administrators who should have additional access permissions defined. Therefore, Alfabet Expand Web cannot be accessed by anonymous users. Furthermore, new users cannot be added to the Alfabet database as anonymous users on their first access. For information about configuring user profiles for anonymous access, see *Defining a User Profile for Anonymous Users* in the reference manual *User and Solution Administration*.
>
> For information about configuring the Alfabet Server to allow anonymous access, see either
>
> - [Access Permissions To Open the Alfabet User Interface](#) in the section [Activating the Dispatch of Email Notifications in Alfabet](#) or
>
> - [Managing Access Permissions for Access from External Applications](#) in the section [Accessing the Alfabet Database with External Applications](#).

The authentication via a company authentication portal can only be performed using the Alfabet Web Application. All other Alfabet components must be configured to use standard login.

Authentication via Federated Authentication

When a user opens the Alfabet user interface via a Web browser, Internet Information Services® send an authentication request to the federated authentication system. Depending on the local configuration, user authentication is done without any user interaction required or the user is provided with a login screen for authentication against the federated authentication system. The Alfabet Web Application impersonates the authenticated user and verifies whether a named user with a user name matching the user's user name already exists in the Alfabet database.

If the user name is found, the user profiles attached to the user will then be displayed and can be selected by the user.

If the corresponding user cannot be found in the Alfabet database, the Alfabet Web Application can be configured either to deny access or to create a Alfabet user of the type Anonymous in the database. If the Alfabet Web Application allows anonymous access, the ReadOnly user profile configured for anonymous access will be displayed. The administrator can later change the **Type** attribute of the newly created anonymous user to `NamedUser` and assign user profiles to the user.

> ⓘ Anonymous access is especially relevant for the configuration of access to Alfabet views from external applications or for links in emails sent via Alfabet.
>
> For Alfabet Expand Web, anonymous access is not suitable because access to configuration functionalities should be limited to solution administrators who should have additional access permissions defined. Therefore, Alfabet Expand Web cannot be accessed by anonymous users. Furthermore, new users cannot be added to the Alfabet database as anonymous users on their first access. For information about configuring user profiles for anonymous access, see *Defining a User Profile for Anonymous Users* in the reference manual *User and Solution Administration*.
>
> For information about configuring the Alfabet Server to allow anonymous access, see either
>
> - [Access Permissions To Open the Alfabet User Interface](#) in the section [Activating the Dispatch of Email Notifications in Alfabet](#) or
>
> - [Managing Access Permissions for Access from External Applications](#) in the section [Accessing the Alfabet Database with External Applications](#).

Alfabet leverages the.Net identity federation mechanism based on SAML 2.0. Tests are conducted on the basis of the OpenAM federation solution.

When a user that is logged in to Alfabet via federated authentication logs out, the user will also be logged out of the SAML system. All other SAML sessions opened for the same session ID in parallel will also be closed. This also applies to multiple instances of the Alfabet user interface opened in different tabs of the same browser.

The authentication via federated authentication can only be performed using the Alfabet Web Application. All other Alfabet components must be configured to use standard login.

Authentication via Client Authentication Certificates

If certificates are used for client/server authentication between the Alfabet Web Application and the Web Browser used to open the Alfabet Web Client, the Alfabet Server can be configured to read the user name of the user from a readable attribute of the certificate. The user name can optionally be constructed from the value of the certificate attribute and a customer defined prefix and/or suffix.

The Alfabet Server will not require the user to enter the user name and password again. The Alfabet Server verifies whether a user of the type Named User with a matching user name already exists in the Alfabet database.

If the user name is found, the user profiles attached to the user will then be displayed and can be selected by the user.

If the corresponding user cannot be found in the Alfabet database, the Alfabet Server can be configured either to deny access or to create a Alfabet user of the type `Anonymous` in the database. If the Alfabet Server allows anonymous access, the ReadOnly user profile configured for anonymous access will be displayed. The administrator can later change the **Type** attribute of the newly created anonymous user to `NamedUser` and assign user profiles to the user.

> ⓘ Anonymous access is especially relevant for the configuration of access to Alfabet views from external applications or for links in emails sent via Alfabet.
>
> For Alfabet Expand Web, anonymous access is not suitable because access to configuration functionalities should be limited to solution administrators who should have additional access

permissions defined. Therefore, Alfabet Expand Web cannot be accessed by anonymous users. Furthermore, new users cannot be added to the Alfabet database as anonymous users on their first access. For information about configuring user profiles for anonymous access, see *Defining a User Profile for Anonymous Users* in the reference manual *User and Solution Administration*.

For information about configuring the Alfabet Server to allow anonymous access, see either

- Access Permissions To Open the Alfabet User Interface in the section Activating the Dispatch of Email Notifications in Alfabet or

- Managing Access Permissions for Access from External Applications in the section Accessing the Alfabet Database with External Applications.

The authentication via a certificate can only be performed using the Alfabet Web Application. All other Alfabet components must be configured to use standard login.

Standard User Login

The standard login should be used for installations that do not allow or cannot support enterprise login. In this case, the user must be authenticated by means of a Alfabet user name and password every time the user tries to access the application. User name and password are evaluated against the Alfabet database.

A named Alfabet user must be created by the user administrator for a successful login process. An initial password should be configured. Passwords are stored encrypted.

Anonymous users cannot access Alfabet components via standard login.

The user's login credentials can be checked against the login data from an external data source instead of maintaining the user data directly in the Alfabet database. The configuration for this setup is described separately in the section Configuring the Alfabet Web Application to Perform User Authentication Based On User Data from an External LDAP Data Source chapter Integrating Data from External Sources.

Standard authentication can be used for login to all Alfabet components.

Customer-Specific Authentication Mechanisms

In addition to the standard authentication mechanisms described above, it is possible to implement customer specific "hooks" for additional customer needs regarding the login of Alfabet users. This might be, for example, an additional existence check on an external server like LDAP or an additional certificate validation on the Web server. Configuration is dependent on the customer requirements and therefore not described in this manual. Customer-specific configurations are not included in a standard Alfabet license and must be purchased separately. Contact Software AG Support for further information.

> ⓘ User attributes such as the email address can be maintained from an LDAP server or an existing database table. This is possible via unidirectional data synchronization between external sources and Alfabet objects. For more information, see the section Integrating Data from External Sources.

Configuring Windows Sign-On

The following steps are required to enable windows sign-on for the Alfabet components:

- Activating Windows Sign-On

- Configuring the Alfabet Web Application to Accept Windows Sign-On

- Configuring a Named User

The following steps are optional in the configuration of enterprise authentication:

- Disabling the Use of the Windows Domain Name in the User Name

- Restricting Authentication to Specific Domains or Specific Users

- Disabling Re-Login with Standard Login Mechanisms

Activating Windows Sign-On

Windows sign-on must be configured for the server alias configuration of the Alfabet Web Application. Windows sign-on is deactivated by default.

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** ✏ button. An editor opens.

4) Go to the **Client Settings** > **Authentication** tab.

5) In the **Mode** field, select `SSO_WinUser` in the drop-down field.

6) Information about the authentication process can be written to a log file. Optionally, you can change the name of the authentication log file in the **Authentication Connection Test Log File** field. If you do not specify a path, the file is located in the physical directory of the Alfabet Web Application. The path specification must be an absolute path. This file is only relevant for a first test of the connectivity. During normal operation the field should be cleared. Make sure that the Alfabet Web Application has write permissions for the file.

7) Click **OK** to save your changes.

Configuring the Alfabet Web Application to Accept Windows Sign-On

To use Windows authentication with the Alfabet Web Application, the authentication mechanisms of the corresponding application directory must be exclusively set to **Integrated Windows Authentication** in the Internet Information Services® Manager. For more information, see the section Configuring Windows Sign-On.

Configuring a Named User

The following section is limited to the basic configuration of a Alfabet user. User data is typically main-
tained in the Administration functionalities of the Alfabet user interface. User data management is available
in the Alfabet Administrator and the user administration is described in detail in the reference manual *User
and Solution Administration*.

> (i) For optional parameters available when configuring a Alfabet user and an explanation of the
> toolbar functionalities available via the **User Management** functionality in the Alfabet Adminis-
> trator, see the section Managing New and Existing Users.

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the
server alias that you want to configure.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via
Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) In the explorer, expand the **User Management** node and click **Users**.

5) In the toolbar, click **New** > **Create User**. An editor opens.

> (i) If enterprise authentication is enabled and a user who is not found in the Alfabet data-
> base attempts to log in, the user will be created automatically in the Alfabet database. If
> the user you want to configure already exists in the Alfabet database, select the user in
>
> the table and click the **Edit** ✎ button.

6) In the editor, go to the **Properties** tab and define the following attributes:

- **Name**: Enter the user's family name. This name is used to identify the user in the Alfabet user
interface.

- **Type**: Select `NamedUser`. User profiles can only be assigned to users of the type `NamedUser`.

- **User Name**: Enter the user's Windows login name in the following format:
[Domain]\[UserName]. You must use a backslash between the domain name and user name.

> (i) The use of Windows domain names as part of the user name can be disabled. If the
> user of the Windows domain names is disabled, enter only the Windows user name in
> the **User Name** field. For more information, see the section Disabling the Use of the
> Windows Domain Name in the User Name.

7) Click **OK** to save the data or click **Cancel** to exit without saving your changes. The new user is
displayed in the table.

8) In the table, select the user and click the **User Profiles...** button in the toolbar. A dialog box opens
that lists all available user profiles.

9) Select all user profiles that you want to assign to the user.

10) Click **OK** to save your changes. The user profile is now attached to the user.

Disabling the Use of the Windows Domain Name in the User Name

When you create an Alfabet user name for Windows sign-on, the Windows domain name is included by default. However, you can remove the domain name from the user name. In this case, every time a user logs in, the domain name will be excluded from the Windows network name for the user.

> ⚠ You should only exclude the domain name if all user names are unique for the enterprise. If user names are only unique for single domains in the enterprise, you should not disable the domain name.

To exclude the domain name when you create the Alfabet user name:

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** ✎ button. An editor opens.

4) In the editor, go to the **Server Settings** > **Security** tab and select the checkbox **Ignore User Domain**.

5) Click **OK** to save your changes.

Restricting Authentication to Specific Domains or Specific Users

Authorization can be configured on a per domain and per user basis. It is possible to restrict access permission for the Alfabet Web Application to specific domains and to specific users within domains.

Domain accessibility configuration is carried out in two steps:

- A general rule specifies whether users from all domains are allowed or not allowed to access the Alfabet Web Application.

- Definitions of domain accessibility configuration allow for the exclusion of specified domains from the general rule. Each domain accessibility configuration of a domain will overwrite the general rule. A domain accessibility configuration for a domain can specify single users to be excluded from the general domain accessibility configuration. For example, you can specify access permissions for single users within the domain when access is denied for that domain.

To configure the general rule for access permissions:

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias that you want to configure.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) In the explorer, click the **Domain Accessibility Configuration** node.

5) In the toolbar, select **Action** > **Allow All Domains** in order to allow all users of all domains that are not specified in an authorization configuration to access the Alfabet Web Application. If you want to deny access to users from domains without the authorization configuration, deselect the option.

> ⓘ A checkmark will be displayed in front of the menu item if the **Allow All Domains** attribute is selected. The **Allow All Domains** attribute is selected by default.

To specify an authorization configuration for a single domain:

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias that you want to configure.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) In the explorer, click the **Domain Accessibility Configuration** node.

5) In the toolbar, click **New** > **Create...**. An editor opens.

6) Edit the following fields:

   • **Name**: Enter the name of the domain.

   • **Allow All Domain Users**: Select the checkbox to allow all users of the domain to access the Alfabet Web Application, if not otherwise specified. If the checkbox is not selected, access to the Alfabet Web Application is denied by default for all users of the domain.

   • **Allow Users**: Specify the Windows user names of the users that the general rule is not valid for. You can only edit this field if you have deselected **Allow All Domain Users**.

   • **Deny Users**: Specify the Windows user names of the users that the general rule is not valid for. You can only edit this field if you selected **Allow All Domain Users**.

7) Click **OK** to save your configuration or click **No** to exit without saving your changes.

Configuring Federated, Portal, or Certificate-Based Authentication

This section informs about the setup required for the Alfabet components to use an existing authentication infrastructure via portal authentication, client authentication certificates or federated authentication for log in to the Alfabet user interface. The implementation of the authentication mechanisms including the required configuration of the Web server depend on the third-party software used for authentication and is not subject of this reference manual.

The following steps are required to enable authentication via a portal, federated authentication, or a client authentication certificate for the Alfabet components:

• Configuring the Alfabet Web Application to Use Authentication via a Portal or Configuring the Alfabet Web Application to Use Authentication via Federated Authentication or Configuring the Alfabet Web Application to Use Authentication via a Client Authentication Certificate

• Configuring a Named User

The following steps are optional in the configuration of enterprise authentication:

• Disabling Re-Login with Standard Login Mechanisms

• Enabling a User to Access Tools for Configuration and Administration

- [Configuring User Authorization Based on Windows Sign-On Data or Data from Federated Authentication](#)

Configuring the Alfabet Web Application to Use Authentication via a Portal

Authentication via a portal must be configured for the server alias configuration of the Alfabet Web Application.

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** button. An editor opens.

4) In the editor, o to the **Server Settings** > **Authentication** tab.

5) In the **Mode** field, select `SSO_Portal`.

6) In the **HTTP Server Variable for Portal Integration** field, enter the name of the server variable corresponding to the HTTP header that contains the user name.

7) Information about the authentication process can be written to a log file. Optionally, you can change the name of the authentication log file in the **Authentication Connection Test Log File** field. If you do not specify a path, the file is located in the physical directory of the Alfabet Web Application. The path specification must be an absolute path. This file is only relevant for a first test of the connectivity. During normal operation the field should be cleared. Make sure that the Alfabet Web Application has write permissions for the file.

8) Click **OK** to save your changes.

Configuring the Alfabet Web Application to Use Authentication via Federated Authentication

Alfabet supports SAML to implement federated authentication. The Alfabet Web Application can act as a SAML service provider.

The following parts of SAML2 are supported:

- Supported bindings:

  - HTTP Redirect

  - HTTP Post

  - SOAP- as back channel for notifications about logout.

- Supported profiles:

  - Web Browser SSO

  - Single Logout

- Supported name identifier formats:

  - Transient identifier

  - Persistent identifier

Configuring the Alfabet Web Application to act as a service provider requires a number of configuration steps that include the server alias configuration and the `Alfabet.config` and `AppSettings.config` file of the Alfabet Web Application:

1) Copy the service provider certificate file and, if applicable, the identity provider metadata file of your SAML solution in a directory accessible for the Alfabet Web Application.

   (i) Note the following:

   - The service provider certificate and certificate password must be XML-conform and must not include any XML reserved characters such as < or >.

   - Certificate signature must be based on SHA-1, SHA-256, SHA-384 and SHA-512.

   - The identity provider metadata can be provided via an URL instead of a file.

   - The file should not be stored in the *config* subdirectory of the Alfabet Web Application.

2) Copy the file `web.sso.saml2.config` from the *Example* subdirectory of the Alfabet Web Application directly to the physical directory of the Alfabet Web Application and change the name of the file to `web.config`.

   (i) If you already configured the `web.config` file of the Alfabet Web Application, you can first change the name of the old `web.config` file to prevent it from being overwritten and then take over all required settings for other features from the old file to the new `web.config` file via a text editor.

3) Open the Alfabet Administrator.

4) Click the **Alfabet Aliases** node in the explorer. A workspace with a toolbar opens.

5) In the table, click the server alias of the Alfabet Web Application. An editor opens.

6) Go to the **Client Settings** > **Authentication** tab.

7) Set the **Mode** attribute to `SSO_FederatedAuthentication`.

8) Click **OK**. The change is saved, and the editor is closed.

9) In the toolbar, click **Tools** > **Configure Alfabet.config**. An editor opens.

10) Click the Browse button on the right of the **Web Folder** field and select the main directory of the Alfabet Web Application from the directory browser. The `Alfabet.config` file in the `config` subdirectory of the selected directory opens in the editor.

11) Check whether the path to the location of the SAML configuration files is correctly defined in the following code:

    ```
    <add key="saml_config_base_path" value="path to the location of the SAML2
    Configuration">
    ```

12) Click **Save**. The change is saved, and the editor is closed.

13) In the toolbar, click **Tools** > **Configure SAML Integration**. An editor opens.

14) Select the path to the physical directory of the Alfabet Web Application via the **Browse** button to the right of the **Path to Web Application Folder** field.

15) Enter the URL for opening the Alfabet user interface in the **Alfabet Web Application URL** field. The URL must start with `https://`.

16) Open one of the following steps according to the requirements of your data and perform the following steps in the selected tab:

- If the identity provider metadata is provided via a file, open the **Configuration File Based** tab.

- If the identity provider metadata is provided via an URL, open the **Configuration Online** tab.

17) Specify the information about the identity provider metadata:

- If you are working in the **Configuration File Based** tab, click the **Select** button on the right of the **Identity Provider Metadata File** field and select the file from the local file system.

- If you are working in the **Configuration Online** tab, enter the URL of the IDP Entity Provider MetaData Export in the **Identity Provider Metadata URL** field.

18) Click the **Select** button on the right of the **Service Provider Certificate File** field and select the service provider certificate file from the local file system.

19) In the **Service Provider Certificate Password** field, enter the password for the certificate.

20) Click the **Generate** button to generate SAML configuration for the Alfabet Web Application.

21) After automatic configuration is performed, a pop-up window will open with information about further configuration requirements. Configure the system as advised in the pop-up window to complete the configuration.

22) Import the service provider certificate file into the Windows® Certificate Store.

> (i) For information about how to perform the import, see the Microsoft® help.

23) On import, the certificate file will get a subject name, serial number, and thumbprint number. This information can be used to identify the file in the Alfabet SAML configuration. You can read the information from the properties of the certificate in the Microsoft® Management Console.

> (i) For information about how to work with the Microsoft® Management Console, see the Microsoft® help.

24) Open the `saml.config` file from the subdirectory ***config*** in the working directory of the Alfabet Web Application in a text editor.

25) Go to the XML element LocalCertificates and substitute the existing sub-element with one of the following:

- To identify the certificate via the subject name:

  ```
  <Certificate Thumbprint="ThumbprintNumberOfCertificate"/>
  ```

- To identify the certificate via the serial number:

  ```
  <Certificate SerialNumber="SerialNumberOfCertificate"/>
  ```

- To identify the certificate via the thumbprint:

  ```
  <Certificate SubjectName="SubjectNameOfCertificate"/>
  ```

26) Open the web.config files and check whether the following settings need adaptation according to your demands:

- If the **External Access** attribute in the **Server Settings** > **Security** tab of the server alias of the Alfabet Web Application is set to **Allowed as Authenticated User**, remove or comment out the following part of the XML content:

```
<location path="ExternalAccess.aspx">

    <system.web>

        <authorization>

            <allow users="*" />

        </authorization>

    </system.web>

</location>
```

- To mark cookies as secure to avoid exposing them over unsecure HTTP channels, add `cookieless="UseCookies"` to the xml element `forms`, like for example:

```
<forms cookieless="UseCookies" loginUrl="Login.aspx" requireSSL="true"
defaultUrl="home.aspx" />
```

27) To export the Alfabet metadata for IdP configuration, open a web browser and call `URLOfTheAlfabetWebApplication /SAML/exportmetadata.aspx`.

Please note that the generation of the SAML configuration via the user interface including the following manual end configuration should be repeated if the following applies:

- Certificates or metadata information have changed on the identity provider side.

- Problems with SAML configuration occur after migration to a new Alfabet release.

It is not possible to re-generate the configuration with a configuration file accessed via URL and having the same name and URL as used for the previous configuration.

On migration from Alfabet 10.5 to Alfabet 10.6, the embedded third-party component Compo-nentSpace® SAML2 for NET4 has been updated. This change requires migration of the current SAML configuration. A migration mechanism is available in the Alfabet Administrator. To migrate the SAML configuration file, click the **Alfabet Aliases** explorer node and in the toolbar of the view that opens, click **Tools** > **Migrate SAML Configuration**.

Configuring the Alfabet Web Application to Use Authentication via a Client Authentication Certificate

Authentication via a Client authentication certification must be configured for the server alias configuration of the Alfabet Web Application.

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** button. An editor opens.

4) In the editor, go to the **Client Settings** tab and open the **Authentication** tab.

5) In the **Mode** field, select `SSO_Certificates` in the drop-down field.

6) Edit the following fields:

- **Certificate Attribute**: Enter the name of the certificate attribute that is used to identify the user.

- **Certificate Value Format**: Select the way that the user name is extracted from the certificate attribute used for authentication:

  - `Parenthesis`: The value defined via the **Certificate Attribute** attibute is scanned for text written in parenthesis. The text in parenthesis is then used as user name, optionally amended by a specified prefix or suffix.

  - `EntireValue`: The value defined via the **Certificate Attribute** attibute is used as user name and may be optionally amended by a specified prefix or suffix.

- **User Name Prefix**: If applicable, enter a prefix that is added to the certificate attribute to generate the login name for the user.

- **User Name Suffix**: If applicable, enter a suffix that is added to the certificate attribute to generate the login name for the user.

- Information about the authentication process can be written to a log file. Optionally, you can change the name of the authentication log file in the **Authentication Connection Test Log File** field. If you do not specify a path, the file is located in the physical directory of the Alfabet Web Application. The path specification must be an absolute path. This file is only relevant for a first test of the connectivity. During normal operation the field should be cleared. Make sure that the Alfabet Web Application has write permissions for the file.

7) Click **OK** to save your changes.

Optionally, you can configure the Alfabet Web Application to protect the certificate files used for authentication via certificates from download via a web browser:

1) Go to the Alfabet Web Application directory in the installation directory of the Alfabet components.

2) Open the `web.config` file in a text editor.

3) Add the following XML element to the XML element `handlers` that is a child of the XML element `system.webServer`:

```
<add name="Test" verb="" path="*.pfx"
type="System.Web.HttpForbiddenHandler"/>
```

Configuring a Named User

The following section is limited to the basic configuration of a Alfabet user. User data is typically maintained in the Administration functionalities of the Alfabet user interface. User data management is available in the Alfabet Administrator and the user administration is described in detail in the reference manual *User and Solution Administration*.

ⓘ For optional parameters available when configuring a Alfabet user and an explanation of the toolbar functionalities available via the **User Management** functionality in the Alfabet Administrator, see the section Managing New and Existing Users.

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer and right-click the server alias that you want to configure and select **Connect**. A login window is displayed.

2) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

3) In the explorer, expand the **User Management** node and click **Users**.

4) In the toolbar, click **New** > **Create User**. An editor opens.

> ⓘ If enterprise authentication is enabled and a user who is not found in the Alfabet database attempts to log in, the user will be created automatically in the Alfabet database. If the user you want to configure already exists in the Alfabet database, select the user in the table and click the **Edit** 🖉 button.

5) In the editor, go to the **Properties** tab and define the following attributes:

- **Name**: Enter the user's family name. This name is used to identify the user as the authorized user of objects.

- **Type**: Select `NamedUser`. User profiles can only be assigned to users of the type `NamedUsers`.

- **User Name**: The user name must match the user name used for the selected single sign-on authentication:

  - If portal authentication is used, enter the user's portal login name.

  - If federated authentication is used, enter the user's federated authentication login name.

  - If authentication via a client side, Client authentication certificate is used, enter the user name resulting from the settings in the server alias configuration. The user name must either correspond to the value of the certificate attribute selected in the server alias configuration or to the value written in parentheses within the certificate attribute. If a prefix and/or suffix is defined for the certificate attribute in the server alias configuration, the value must be amended with the specified prefix and/or suffix.

6) Click **OK** to save the data or click **Cancel** to exit without saving your changes. The new user is displayed in the table.

7) In the table, select the user and click the **Profiles...** button in the toolbar. A dialog box opens that lists all available user profiles.

8) Select all user profiles that you want to assign to the user.

9) Click **OK** to save your changes. The user profile is now attached to the user.

Configuring Standard Login

When standard login is used, users log in to Alfabet with an Alfabet specific user name and password.

Before a user can access Alfabet, you must configure at least one user profile that specifies the functionalities that may be accessed by the user. User profiles and the modules assigned to them are defined in the configuration tool Alfabet Expand. For more information, see the section *Making Functionalities Accessible to a User Profile* in the reference manual *Configuring Alfabet with Alfabet Expand*.

The following steps are required to enable standard login for the Alfabet components:

- Activating Standard Login

- Configuring the Alfabet User

- Defining User Passwords

The following steps are optional in the configuration of the standard login:

- Enforcing Passwords and Defining Rules for Password Specification

- Defining Password Expiration

Activating Standard Login

Standard login must be activated for the server alias configuration of the Alfabet components. Standard login is activated, by default.

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** 🖊 button. An editor opens.

4) In the editor, go to the **Client Settings** tab and open the **Authentication** tab.

5) In the **Mode** field, select `Standard` in the drop-down field.

6) Click **OK** to save your changes.

Configuring the Alfabet User

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias that you want to configure.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) In the explorer, expand the **User Management** node and click **Users**.

5) In the toolbar, click **New** > **Create User**. An editor opens.

6) In the **Properties** tab, define the following attributes:

- **Name**: Enter the user's family name. This name is used to identify the user as the authorized user of objects.

- **Change Password**: If you select this checkbox, the user must define his/her password during the first login. After the user has changed his/her password, this option will automatically be reset to deactivated. If you deselect the checkbox, you should assign a password to the user. Otherwise, the user will be able to log in without a password.

  > (i) For more information about assigning a password to an Alfabet user, see Defining User Passwords.

- **Type**: Select `NamedUser`. User profiles can only be assigned to named users.

- **User Name**: Enter a user name for the user.

7) Click **OK** to save the data or click **Cancel** to exit without saving your changes. The new user is displayed in the table.

8) In the table, select the user and click the **Profiles...** button in the toolbar. A dialog box opens that lists all available user profiles.

9) Select all user profiles that you want to assign to the user.

10) Click **OK** to save your changes. The user profile is now attached to the user.

The new user is displayed in the table with the checkmark displayed in the **Alfabet Managed User** column (=`TRUE`). This means that the user logs in with user name and password and that the user password is managed in Alfabet. The user will be created without a random password that is stored encrypted. The user will not be able to login unless you use one of the mechanisms described below to re-set the password and communicate the new password to the user. Please note that the actions for managing user passwords are only performed for Alfabet internal users.

Defining User Passwords

This section informs about different ways to set an initial password for a user and how to reset an existing user password.

When you create a new Alfabet user, the user is created with a random password that is stored encrypted and the user will not be able to log in. Setting passwords requires one of the actions described below. These mechanisms can also be used to re-set a user password or to create a new password for users after the user's password have been cleared via the Action > Clear User Password or Action > Clear o see which users currently do not have a password defined, select the **Show Internal Users Without Password** checkbox in the **User Management** > **Users** view of the connected server alias.

- Assigning a Password via System-Generated Email Notifications

- Assigning a Password without System-Generated Emails Involved

- Changing an Existing Password

- Clearing an Existing Password

Assigning a Password via System-Generated Email Notifications

After having created the user, you can use the **Regenerate Password** function to trigger sending two emails to the new user:

- An email informing the user about the user name specified for login and a link to the login screen for first login.

- A second email containing an automatically generated password.

Immediately after having entered the user name and password in the login screen for first login, the user is prompted to change his/her password.

> (i) This functionality includes sending emails to the user via the system. Make sure that system emails are activated for your Alfabet installation. For more information about activating sending of emails, see the section Activating the Dispatch of Email Notifications in Alfabet.

The **Regenerate Password** function can be used to change the password for one, all or a subset of existing users:

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias that you want to configure.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) In the explorer, expand the **User Management** node and click **Users**.

5) In the table, select the user for which you want to initialize login and do one of the following.

- To send a password via email to one or multiple selected users, select the users in the table and select **Action** > **Regenerate Password** in the toolbar.

- To send a password via email to all Alfabet internal users that currently have no password assigned, select **Action** > **Regenerate Empty Passwords** in the toolbar.

- To send a password via email to all Alfabet internal users, select **Action** > **Regenerate All Passwords** in the toolbar.

- To send a password via email to a user that is currently not able to log in because the maximum number of failed login counts has been exceeded, select Action Reset Failed Login Count and Regenerate Password. The number of failed login counts will be reset, the lock will be removed, and the user will be able to log in with the new password.

> (i) The Alfabet Web Application can be configured to display a link that the user can click to reset the password if he/she has forgotten the password. If the user clicks the link, the **Regenerate Password** function will be invoked automatically. For information about the required configuration of the Alfabet Web Application, see Adding a Link to Request a New Password If Users Forget Their Passwords.

Password regeneration actions are logged in a log file. This is by default the `PasswordChangeLog.txt` file in the `Runtime\Tmp` subdirectory of the physical directory of the Alfabet Web Application. For information about how to change the location of the log file, see Defining the Location of the Log File for Password Reset and Regeneration Actions.

Assigning a Password without System-Generated Emails Involved

The **Change Password** setting for the user should also be activated to force the user to change his/her password during first login. This activation must be performed after having reset the password:

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias that you want to configure.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) In the explorer, expand the **User Management** node and click **Users**.

5) In the table, select the user for which you want to initialize login.

6) In the toolbar, select **Action** > **Change Password** to change the selected user's password.

7) In the window that opens, enter the old password in the **Current Password** field, enter the new password in the **New Password** field, and re-enter the new password in the **Confirm New Password** field.

8) Click **OK** to save your changes.

9) In the toolbar, click **Edit** ✐. An editor opens.

10) In the **Properties** tab, select the **Change Password** checkbox.

> ⓘ The checkmark in the **Change Password** checkbox is automatically removed when the password has been changed by the user.

11) Click **OK** to save your changes.

12) Inform the user about his/her user name and password and the web address for access to the Alfabet user interface.

Changing an Existing Password

Users with the appropriate rights can change their password in Alfabet in the **<UserName>** menu in the upper-right corner of the Alfabet screen.

> ⓘ For users logging in by means of an enterprise login, the **Change Password** function in the **<UserName>** menu is disabled.

You can force a user to change his/her password on the next login attempt:

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias that you want to configure.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) In the explorer, expand the **User Management** node and click **Users**.

5) In the table, select the user for which you want to initialize login.

6) In the toolbar, click **Edit** . An editor opens.

7) In the **Properties** tab, select the checkbox **Change Password**.

> The checkbox is automatically reset to deselected when the user has changed the password.

8) Click **OK** to save your changes.

9) The user will be asked to change his/her password on next login to the Alfabet user interface.

Clearing an Existing Password

You can reset the password of one or multiple users without knowing the current password:

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias that you want to configure.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) In the explorer, expand the **User Management** node and click **Users**.

5) Do one of the following:

- To change the password for a single user or a sub-group of users, select the users for whom you want to reset the password in the table and select **Action** > **Clear Password** in the toolbar.

- To change the password for all users, select **Action** > **Clear All Passwords** in the toolbar.

6) The current password(s) will be removed. Confirm the warning that follows by clicking **OK**.

After having reset the password, a new password should be defined for the user using one of the methods described in the sections Assigning a Password via System-Generated Email Notifications and Assigning a Password without System-Generated Emails Involved.

> User passwords are stored encrypted in the Alfabet database. If a user password is reset, the password will be replaced in the Alfabet database with an encrypted value representing NULL. This ensures that the absence of the user password is not visible on the database level.

Password reset actions are logged in a log file. This is by default the `PasswordChangeLog.txt` file in the `Runtime\Tmp` subdirectory in the physical directory of the Alfabet Web Application. For information about how to change the location of the log file, see Defining the Location of the Log File for Password Reset and Regeneration Actions.

Enforcing Passwords and Defining Rules for Password Specification

The server alias configuration includes parameters that restrict the definition of passwords to a defined syntax. For example, you can require the user to use special characters in the password or to use a password of a defined minimum or maximum length.

Enforcing password definition rules will also disable the ability to log in without a password. A nonexistent password will not match any user password criteria settings. The user will therefore be prompted on first login attempt to define a password matching the defined password criteria. Users with a password not matching the defined criteria will also be prompted to change their password during login.

To define password criteria:

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** 🖉 button. An editor opens.

4) In the editor, go to the **User Password Settings** tab and select the checkbox **Enforce Password Criteria**.

5) Edit the following parameters, according to your enterprise's password requirements:

- **Minimum Password Length**: The minimum number of characters required for a user password.

- **Recent Passwords Number**: The number of recent passwords. If a user password is changed, the old password will be stored in addition to the current one. User passwords will be stored according to the specified number of recent passwords. For example, if the number of recent passwords is 4, the current and the last three used passwords are stored. If a user changes the password, he/she cannon reuse a stored password.

  > ⓘ The list of recently used passwords can be deleted from the Alfabet database for one or multiple selected users via the **Action** > **Clear Recent Passwords** option in the **User Management** functionality available via the connected server alias. This will allow the user to use old passwords that have been used in the past.

- **Min. Lower Case Letters**: The minimum number of lower-case letters required in a password.

- **Min. Upper Case Letters**: The minimum number of upper-case letters required in a password.

- **Min. Digits**: The minimum number of digits required in a password.

- **Min. Special Characters**: The minimum number of special characters required in a password.

6) Click **OK** to save your changes.

Defining Password Expiration

There are two mechanisms to force the Alfabet user to redefine his/her user password:

- All users have to change their passwords in a specified interval.

- On a per user basis, a password expire date can be specified.

The following information is available:

- [Defining a Change Password Interval for All Users](#)

- [Defining a Password Expire Date for a Single User](#)

Defining a Change Password Interval for All Users

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** button. An editor opens.

4) In the editor, go to the **User Password Settings** tab and in the **Interval** field, enter the number of days for which the password is valid. After the defined time ends, the user will be prompted to select a new password. Enter "-1" to configure the setting so that the user will never be required to change the password.

5) Click **OK** to save your changes.

Defining a Password Expire Date for a Single User

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias that you want to configure.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) In the explorer, expand the **User Management** node and click **Users**.

5) In the table, select the user and then click the **Edit** button in the menu.

6) In the editor that opens, go to the **Properties** tab and enter an expiration date for the password in the **Password Expiration Date** field.

7) Click **OK** to save your changes.

Adding a Link for Request of User Credentials to the Login Page

A link **Click here to request access credentials** can be displayed on the login screen. New users that would like to have access to Alfabet but do not yet have a user name and password assigned, can click the link that will lead them either to a URL for web-based request of user credentials or will open an email to a predefined email addressed to the system administrator granting access to Alfabet.

The link for request of access credentials will only be displayed if the feature is activated in the server alias of the Alfabet Web Application:

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** button. An editor opens.

4) Go to the **Client Settings** > **Authentication** tab.

5) In the **Request Credentials URL** field, define the link target of the **Click here to request access credentials** link:

- If a URL is defined in the field, the link on the login screen will open the defined URL. The URL must be defined starting with http:// or https://.

- If an email address is defined in the field, the link in the login screen will open an email to the defined email address with the default mail client of the user with the Subject line **Access Credential Request**. The email address must be defined as `mailto:` followed by the email address.

Adding a Link to Request a New Password If Users Forget Their Passwords

The Alfabet Web Application can be configured to provide an **I forgot my password** link to the user in the login screen. If the user clicks the link, the user password will automatically be reset, and the **Regenerate Password** function will be invoked for the user. The user will receive two emails:

- An email informing the user about the user name specified for login and a link to the login screen for first login.

- A second email containing an automatically generated password.

Immediately after having entered the user name and password in the login screen for the first login, the user will be prompted to change his/her password.

Note the following about this functionality:

- This functionality includes sending emails to the user via the system. Make sure that system emails are activated for your Alfabet installation. For more information about activating sending emails, see the section Activating the Dispatch of Email Notifications in Alfabet

- If the user has exceeded the maximum number of failed login attempts, the link will be deactivated, and the user will be prompted to contact the user administrator to re-activate login. For more information about blocking user log in after a maximum number of failed login attempts, see Limiting the Number of Failed Login Attempts

- After having clicked the link, the user will be informed that he/she will receive the information about the new password via email. This information is displayed regardless of the availability of a user with the current user name in the database to avoid that unauthorized persons can find out whether an entered user name exists in the Alfabet database.

Each time a password is regenerated via email, either by a user administrator in the **Users Administration** functionality or because the user clicks the **I forgot my password** link on the login screen, a counter is incremented. The counter is reset to zero on login of the user with the last regenerated password. If the counter reaches a configured maximum number, the password regeneration functionalities are deactivated for the user unless the counter is reset via the **Action** > **Reset Regenerated Passwords Counter** option.

To configure the Alfabet Web Application to provide a link to reset a forgotten password on the login screen:

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** ✏ button. An editor opens.

4) Go to the **User Password Settings** tab.

5) Select the **Enable Forgot Password** checkbox.

6) Enter the number of allowed unused password regeneration requests into the **Maximum Number of Regenerated Passwords** attribute.

7) Click **OK** to save your changes.

Defining the Location of the Log File for Password Reset and Regeneration Actions

Resetting and regenerating user passwords can be performed for multiple users or all users in the Alfabet database. If the action fails for individual users (for example, because a password could not be regenerated for a user because no email address was defined for the user), it will nevertheless be performed for the re-maining users. The results of the actions for resetting and regenerating user passwords are written to a log file. This log file is by default located in the `Runtime\Tmp` subdirectory of the physical directory of the Alfabet Web Application and is named `PasswordChangeLog.txt`.

You can change the name and location of the log file in the alias configuration of the Alfabet Web Application:

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** ✏ button. An editor opens.

4) In the editor, go to the **User Password Settings** tab and enter the absolute path to and name of the new log file in the **Reset/Regenerate Password Log File** field.

5) Click **OK** to save your changes.

Disabling Re-Login with Standard Login Mechanisms

By default, the **<User Name>** menu in the upper-right corner of the Alfabet user interface and in Alfabet Expand Web provides a **Log out** option.
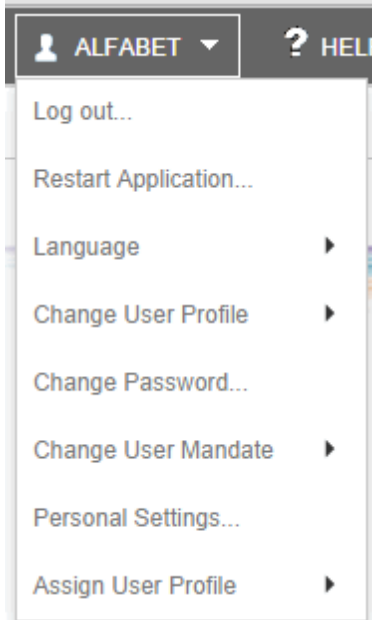
If the user selects **Log out**, the session is closed, and a logout screen is displayed with a link for re-login. The link opens a login screen. This login screen allows the user to log in with a user name and password using standard login. If single sign-on is used for login, a link that triggers re-login via the single sign-on mechanism is also displayed.

If single sign-on is used for login, user authentication will be executed using the external single sign-on mechanisms. During login, the Alfabet Server only uses the Alfabet database to evaluate whether the user with the given login name exists in the Alfabet database, whether the user is an anonymous or named user, and which user profile is assigned to the user. The single sign-on login password is not read from the Alfabet database.

Therefore, a user that is logged in to Alfabet via single sign-on cannot automatically re-login to the Alfabet user interface using the standard login included in the re-login options that are displayed by default. There are three options to handle the re-login option of Alfabet:

- To allow a user to re-login to Alfabet with his/her single sign-on login credentials, the single sign-on login password of the user must be manually added to the configuration of the user in the Alfabet database. For more information about configuring user passwords for standard login, see Defining User Passwords.

- To allow the user to re-login to Alfabet with his/her single sign-on user name and an Alfabet-specific password, a Alfabet-specific password must be manually added to the configuration of the user in the Alfabet database. For more information about configuring user passwords for standard login, see Defining User Passwords.

- If users should not have the option to re-login to Alfabet with standard login authentication, the **Log out** option in the **User** menu can be configured not to provide standard login in the login screen but only the link to re-login via single sign-on.

> If a user logged in via single sign-on cannot be found in the Alfabet database, the Alfabet Web Application can be configured either to deny access or to create an Alfabet user of the type `Anonymous` in the Alfabet database. A re-login via the login screen is only possible with a user name already available in the Alfabet database because the new user is created with a random password. To allow the user to re-login with user name and password, the user password must be re-set and a new password must be assigned to the user by a user administrator. For more information, see Defining User Passwords.

To configure the **Log out** option in the **User** menu for log out without displaying a standard login option on the login screen:

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** button. An editor opens.

4) In the editor, go to the **Server Settings** > **Security** tab and deselect the checkbox **Allow Re-Login**.

5) Click **OK** to save your changes.

Changing the Login Mode from Single Sign-On or LDAP to Standard Login

Users that are created via login using single sign-on mechanisms or authentication via an external LDAP server are regarded as Alfabet external users. The functionalities available for standard login such as resetting and regenerating passwords are not applied to external users. If you create a user directly in Alfabet, the user will be an internal user and standard login can be configured for the user.

Whether the user is an internal or external user is stored in the **Alfabet-Managed User** attribute of the user. The **Alfabet-Managed User** attribute is set to `False` when a user is created via single sign-on mechanisms and set to `True` if the user is created by a user administrator in the **Users Administration** functionality.

You can convert a user created via single sign-on mechanisms or via external LDAP login to an Alfabet internal user:

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias that you want to configure.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) In the explorer, expand the **User Management** node and click **Users**.

5) In the table, select the user or multiple users that you want to convert to internal users.

6) In the toolbar, click **Action** > **Set as Alfabet-Managed User**.

7) Confirm the warning by clicking **Yes**.

There is no mechanism to change an Alfabet internal user to an external user because the Alfabet settings required for external authentication cannot be regenerated for internal users. If you want an internal user to be managed via single sign-on or an external LDAP source, you must delete the user and the user must be recreated via a first login action.

Enabling a User to Access Tools for Configuration and Administration

A user can only execute Alfabet batch processing applications or work with the configuration tool Alfabet Expand or the Alfabet Diagram Designer, if the user is of the type `NamedUser` and permission for access to the tools is explicitly granted to the user in the user data configuration in Alfabet. For Alfabet Expand access can additionally be restricted per user to a subset of the licensed functionalities.

Do the following to define access permissions for Alfabet Expand or Alfabet batch processing applications for a user:

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias that you want to configure.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) In the explorer, expand the **User Management** node and click **Users**.

5) In the table, select the user and then click the **Edit** ✏️ button in the menu.

6) In the editor that opens, go to the **Permissions** tab and select the **Can Execute Batch Jobs** checkbox or **Has Access to Diagram Designer** checkbox to grant the respective permission.

7) In the **Expand Permissions** tab, select the **Has Access to Alfabet Expand** checkbox and check all functionalities for Alfabet Expand that shall be available to the user.

8) Click **OK** to save your changes.

Enabling a User to Execute Alfabet RESTful Service Calls

A RESTful API is available for the Alfabet application that provides easy access to the content in the Alfabet database. Service calls to the Alfabet RESTful API of the Alfabet Web Application are only processed by the Alfabet Web Application if an active license for the Alfabet Data Integration Framework (ADIF) is available and the Alfabet Web Application is configured to accept the calls.

In addition, the service calls must be sent with an authorization for an Alfabet user with the required access rights to execute the call to the relevant end-point of the Alfabet RESTful services.

> ⓘ For information about how to implement the Alfabet RESTful services and a complete overview of the authentication mechanisms required to send RESTful service calls, see the reference manual *Alfabet RESTful API*.

Do the following to define access permissions for the Alfabet RESTful services for a user:

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias that you want to configure.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) In the explorer, expand the **User Management** node and click **Users**.

5) In the table, select the user and then click the **Edit** ✎ button in the menu.

6) In the editor that opens, go to the **Permissions** tab and select the **Has API V2 Access** checkbox.

7) Specify details about the access permissions with the following attributes:

- **API V2 Token Duration (minutes)**: The RESTful service interface on the client side must be implemented to send a request for an authorization code prior to sending a data request. The authorization code received in the response of the authorization request can be used in data request calls to the Alfabet RESTful API posted within a limited amount of time (per default 20 minutes) after receiving the authorization code. The time period of the validity of the authorization code can be changed via this attribute. Enter the number of minutes the authorization code shall be valid.

- **API V2 Access Options**: By default, most of the options are checked in this field to give the user access to all central functionality provided for the Alfabet RESTful services. Deselect all options that the user should not have permissions to perform:

    - **Has Meta Model Access**: If selected, the user has access to the REST endpoints `metamodel`, `classes` and `enums` to read information about the structure of the Alfabet class model including enumerations and culture settings.

    - **Has GetObjectsByRefs Access**: If selected, the user has access to the REST endpoint `objects` to read information about data stored for objects in the Alfabet database that are found via specification of the object's `REFSTR` in the REST API request. Please note that access to object data is also controlled by per object class and per object permissions.

    - **Has GetObjectsByReport Access**: If selected, the user has access to the REST endpoint `objects` to read information about data stored for objects in the Alfabet database that are found via a configured report. Please note that access to object data is also controlled by mandate settings. In addition, access permission to the configured report must be granted via the user profile configuration and the attribute settings of the configured reports.

    - **Has GetObjectsByFilter Access**: If selected, the user has access to the REST endpoint `objects` to read information about data stored for objects in the Alfabet database that are found via specification of the object class and filter conditions in the call. Please note that access to object data is also controlled by per object class and per object permissions.

    - **Has CreateObjects Access**: If selected, the user has access to the REST endpoint `update` to create new object in the Alfabet database. Please note that object class specific permissions must also be granted.

    - **Has UpdateObjects Access**: If selected, the user has access to the REST endpoint `update` to update data for existing objects in the Alfabet database. Please note that object class specific and object specific permissions must also be granted.

    - **Has DeleteObjects Access**: If selected, the user has access to the REST endpoint `delete` to delete existing objects in the Alfabet database. Please note that object class specific and object specific permissions must also be granted.

    - **Has AnonymizeUser Access**: If selected, the user has access to the REST endpoint `anonymizeuser` to anonymize users that are found via specification of the user's `REFSTR` in the REST API request in the Alfabet database.

- **Has ADIFAPIInvocationAccess**: If selected, the user has access to the REST endpoints `adifimport` and `adifexport` to trigger execution of ADIF jobs based on an ADIF scheme in the Alfabet database.

- **Has WorkflowAPIInvocation Access**: If selected, the user has access to the REST endpoint `workflow` to trigger start of a workflow based on a workflow template in the Alfabet database.

- **Has MonitoringAPI Access**: If selected, the user has access to the REST endpoint monitor to check whether the Alfabet components are available.

8) Click **Generate API Password**. A code is generated and stored in the Alfabet database.

9) Copy the code and store the information about user name and code for use on client side.

10) Click **OK** to save your changes and to close the editor.

11) If the user should be used to execute Alfabet RESTful service calls for self-reflective events, select the user in that table and click **Action** > **Set as User Allowed to Execute Self-Reflective Events** in the toolbar.

> (i) Only one user can be used to execute self-reflective events. If you set a user as allowed to execute self-reflective events while another user has already been selected for the task, the setting for the original user will be removed and the task will be moved to the current user.
>
> For information about self-reflective events, see *Configuring Events* in the reference manual *Configuring Alfabet with Alfabet Expand*.

Allowing a User Administrator to Change His/Her Own Access Data

The user administrator with access to the **Users Administration** functionality does not necessarily have access to, for example, configuration and batch tools or to the RESTful service interface. To prevent a user administrator from granting himself/herself elevated access permissions, the default configuration of Alfabet Web Application prohibits the user administrator to edit /his/her own data in the **Users Administration** functionality as well as in the user's object profile that is accessible via the **Users Administration** functionality.

If the user administrator should have access to his/her own user data, this restriction can be deactivated in the server alias of the Alfabet Web Application:

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** 🖉 button. An editor opens.

4) In the editor, go to the **Server Settings** > **Security** tab and select the **Enable Changes To Own Authorization** checkbox.

5) Click **OK** to save your changes.

Configuring the Alfabet Web Application to Accept Self-Signed Certificates for Integration Solutions

For integration solutions based on web services (such as ARIS- Alfabet Interoperability Interface, import of data from Technopedia®, or Jira® integration), self-signed certificate validation can be used on HTTPS connections. This is independent of the configuration of the authentication mode for the Alfabet Web Application.

The following configuration is required:

1) Copy the self-signed certificates from the third-party web service to a local folder that the Alfabet Web Application has access permissions to.

2) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

3) In the table, click the server alias that you want to configure.

4) In the toolbar, click the **Edit** ✎ button. An editor opens.

5) Open the **Server Settings** > **Security** tab and enter the path to the folder in the **Path for Self-Signed Public Certificate Files** field.

Limiting the Number of Failed Login Attempts

The maximum allowed number of failed login attempts is configurable in the server alias in the **Server Settings** > **Security** > **Maximum Number of Failed Logins** attribute. If the attribute is set to-1, the number of consecutive failed login attempts is not limited. If the attribute is set to a positive integer, the number of consecutive failed login attempts will be counted. The counter for failed login attempts is reset to zero after a successful login. When the failed user login attempt equals the configured maximum number, the user will not be able to log in any more. To enable the user to log in again, the user administrator must reset the count for failed login attempts via the **Action** > **Reset Failed Login Count** option in the toolbar of the **User Administration** functionality. The number of failed login attempts is displayed to the user administrator in the **User Administration** functionality. If a user has reached the number of failed consecutive login attempts, the login count will be highlighted red.

⚠ A restriction of the maximum number of consecutive failed login attempts is not possible if authentication is managed via an external data source such as an LDAP server. For more information about authentication via an external data source, see Integrating Data from External Sources.

If a user logs in to an Alfabet database with different Alfabet components, failed login will only be counted if the server alias used for login is configured to restrict the number of failed logins. If a user attempts to log in with one Alfabet component and the attempt fails and the user then attempts to log in to another Alfabet component and the attempt fails, this is registered as two consecutive failed login attempts.

Entering a wrong current password on attempt to change a password is also counted as a failed login attempt. If the number of failed login attempts is reached while a user currently logged in to Alfabet changes his/her password, the user will be logged out and will not be able to re-login.

When a user is blocked because the maximum number of consecutive failed login attempts has been exceeded, the user administrator must re-set the count of failed login attempts via an administrative user profile in the Alfabet user interface or via the Alfabet Administrator to enable login of that user:

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias that you want to configure.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) In the explorer, expand the **User Management** node and click **Users**.

5) In the table, select the user for which you want to re-set the login count.

6) In the toolbar, click one of the following:

   - **Action** > **Reset Failed Login Count** to reset the login count only.

   - **Action** > **Reset Failed Login Count and Regenerate Password** to reset the login count and change the password of the user to a new generated password automatically sent via email to the user.

Tracking User Login

Information about the login and logout of user to Alfabet components is available via the following methods:

- [Tracking Login Actions in the Windows Event Log](#)

- [Tracking Login to the Alfabet User Interface per User and User Profile](#)

- [Reading Login Relevant Data from the Alfabet Database](#)

Tracking Login Actions in the Windows Event Log

The Alfabet components can be configured via the server alias to write information to the Windows Event Log. The following are written to the event log: the registration in the event log, shutdown of the Alfabet Web Application and the reason for shutdown, and user log in and log out. Each Alfabet component that shall write information to the Windows Event Log must be configured to use Event Logging and must be registered with the Windows Event Log as described below. In the event log, the server alias name is displayed as **Source** unless otherwise specified in the server alias configuration. If multiple Alfabet components are involved in a login action and both are configured to write messages to the Windows Event Log, both components will write a separate message for the action. Management of authentication via data synchronization with an external LDAP server may also result in multiple entries per login.

When a user logs in or out of the Alfabet components, the user name and the information whether login or logout was successful will be logged. This does not only include login/logout to the Alfabet user interface but all processes that require a login such as sending requests to the Alfabet RESTful API.

If the **Server Settings** > **General** > **Update History User Name** attribute of the server alias is set to TECH_NAME, the **Technical Name** defined for the user will be used in the Windows Event Log instead of the user's name. It is essential to define the **Technical Name** attribute for all users if this setting is used. The **Technical Name** can only be used in the Windows Event Log if authentication is performed via the user data in the Alfabet database.

The following action is required to enable writing messages to the event log:

1) Run the Alfabet Administrator as administrator.

2) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

3) In the table, click the server alias that you want to configure.

4) In the toolbar, click the **Edit** button. An editor opens.

5) In the editor, go to the **Server Settings** > **Logging** tab and set the following attributes:

   • **Activate Logging**: Select the checkbox.

   • **Windows Event Logging** (Log-in/-out): Optionally, alter the name that is used as **Source** in the Windows Event Log in the **Event Source Name** parameter. By default, the **Event Source Name** parameter is the server alias name.

6) Go to the **Server Settings** > **Logging Details** tab.

7) In the table, set the following in the row for the **Subject** `UserLogon`:

   • Click the cell of the **Event** column. An **x** will be displayed, and the Windows event logging will be activated.

8) Click **OK** to save your changes and close the editor.

9) In the explorer of the Alfabet Administrator, right-click the server alias and select **Register Event Logger** in the context menu.

10) Click **OK** to save your changes and close the Alfabet Administrator.

11) Open the Windows Event log and check whether the registration is logged for the specified **Source**.

The registration of the Event Logger must be re-executed with the Alfabet Administrator run as administrator after each change to the configuration described above.

Tracking Login to the Alfabet User Interface per User and User Profile

A user login tracking function is optionally available that tracks user login actions by writing the login information to a database table in the Alfabet database. If the functionality is activated in the server alias configuration of the Alfabet Web Application, an entry is written to the database table `ALFA_USERLOGIN_TRACKING` whenever a user logs in to the Alfabet user interface or changes the user profile that she/he is logged in with.

Any other logins will not be logged (for example, accessing the Alfabet Web Application via Alfabet RESTful services, running a batch job, or logging in to Alfabet Expand). An exception to this is when the Alfabet user interface is run from Alfabet Expand and the server alias of Alfabet Expand is either configured to perform user login tracking or configured to use an Alfabet Web Application tracking user login as the test application.

The `ALFA_USERLOGIN_TRACKING` table has the following columns:

• `LOGIN_TIME`: The date and time at that the user has logged in.

• `USER_NAME`: The user name of the user that logged in.

• `USER_PROFILE`: The user profile that the user logged in with.

- `PROFILE_TYPE`: The profile type provides information about the access permissions granted to the user. If the user profile is set to ReadOnly, the user profile type will be set to `Viewer`, if the user profile grants ReadWrite access permissions, the user profile type will be set to `Functional`. If the user has logged in with an administrative user profile that grants extended access permissions, the user profile type will be `All`.

This database table is not part of the Alfabet meta-model and therefore will not be visible to customers in the Meta-Model or the ADIF tab in Alfabet Expand. The data from the table is not displayed in any standard Alfabet view. To create a report about the login data, you can either use an external reporting tool with direct access to the Alfabet database or create a configured report based on native SQL with the functionalities for creating configured reports available in Alfabet Expand.

> For information about creating a configured report based on native SQL, see the section *Configuring Reports* in the reference manual *Configuring Alfabet with Alfabet Expand*.
>
> For information about accessing the Alfabet database with external reporting tools, see the section [Accessing the Alfabet Database with External Applications](#).
>
> For information about making the configured report or the external report available to a user in the Alfabet user interface, see the section *Integration of Configured Reports in the Alfabet Interface* in the reference manual *Configuring Alfabet with Alfabet Expand*.
>
> For information about restricting access to the configured report to defined users, see the section *Defining and Managing User Access to Configured Reports* in the reference manual *User and Solution Administration*.

> ⚠ Prior to activating the functionality, you should check that storage of the data is compliant with the local data protection acts and policies because sensitive user data is stored.

Activation of login tracking is done in the server alias configuration of the Alfabet Web Application. Perform the following configuration steps with the configuration tool Alfabet Administrator:

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** 🖉 button. An editor opens.

4) In the editor, go to the **Server Settings** > **General** tab and enable the **Track User Login** option.

5) Click **OK** to save your changes.

Reading Login Relevant Data from the Alfabet Database

The Alfabet object class User Login Details (`ALFA_USERLOGIN_DETAILS`) stores login relevant user data like password settings and information about the last login of the user. This does not include a login history. Login information is limited to the last login of a user. An audit history can be activated for the table to track the login history.

Login to any Alfabet component is taken into account. This includes login to the Alfabet Web Application via the Alfabet RESTful services. For the Alfabet RESTful services, request of the user token is regarded as login while all subsequent requests with the same user token do not change the data in the user login details.

The object class is visible in the class model available in Alfabet Expand. Configured reports can be built according to demand.

> For information about creating configured reports, see *Configuring Reports* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- USER: A reference to the user that the data belongs to.

- ALFA_LICENSE_POLICY: This property is currently not used. It is implemented for future use.

- PASSWORD: The user's current password. The password is stored encrypted.

- PASSWORD_ID: The password identification number. The password id is stored encrypted.

- PWDLASTUPDATE: The date when the password was last changed.

- PASSWORDTIME: This property is a technical property and not relevant.

- PWDEXPIREDATE: The date specifying when the password expires.

  > Password expiration dates are set by user administrators. For more information, see Defining Password Expiration.

- CHANGEPASSWORD: If set to 1, the user must change the password upon the first login. If set to zero, no password change is required.

  > Whether a user must change the password upon the first login is managed by user administrators. For more information, see Changing an Existing Password.

- FAILED_LOGIN_COUNT: The number of failed logins.

  > Users can be excluded from login after a configured number of failed login attempts. For more information, see Limiting the Number of Failed Login Attempts.

- PAT: This property is currently not used.

- LASTLOGIN: The time the last login occurred.

- LAST_LOGIN_IP: The IP address of the last user login. The IP address will only be written into the table for login to the Alfabet user interface, login to the Alfabet Web Application via the Alfabet RESTful services and login to Alfabet Expand Web.

- LASTUSERPROFILE: The name of the user profile that the user was last logged in with.

- LOGIN_SESSIONID: The session identifier used for the last successful login.

- USAGETERMSCONFIRMATIONTIME: The date and time when the user has acknowledged the usage terms of the application. This attribute will only be filled if a login acknowledgement screen is configured in the GUI scheme configuration. If configured, the content of a configurable URL, containing for example the terms of use for the product, will be displayed to the user after successful login and the user will be mandated to click the acknowledgement button before the actual content of the Alfabet application is shown.

> ⓘ For more information about configuring an acknowledgement screen see *Configuring GUI Scheme Definitions for the Alfabet Interface* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- `LASTUSERCULTURE`: The culture the user last used when rendering the user interface.

- `LOGIN_MESSAGE`: If user login failed, this attribute stores the message returned to the user upon last attempted login.

- `LOGIN_SOURCE`: The Alfabet Component the user last logged in to It will be set to one of the following:

  - `WebApplication` for login via the Alfabet user interface.

  - `ExpandWeb` for login to Alfabet Expand Web.

  - `ExpandWindows` for login to Alfabet Expand Windows

  - `BatchJob` for execution of batch jobs.

  - `Admin` for log in via the Alfabet Administrator.

  - `REST_API` for access to the Alfabet database via the Alfabet RESTful services.

- `PWD_REGENERATION_COUNT`: The number of password regeneration emails sent out to the user without being used for login by the user.

> ⓘ For more information about password regeneration mechanisms, see Assigning a Password via System-Generated Email Notifications and Adding a Link to Request a New Password If Users Forget Their Passwords.
>
> Users can be excluded the password regeneration functionalities after a configured number of unused password regenerations. For more information, see Adding a Link to Request a New Password If Users Forget Their Passwords.

Configuring the Alfabet Server to Control Client Authentication Settings

> When the Alfabet Server is configured to control authentication, the remote alias configurations of the relevant clients must be available both in the `AlfabetMS.xml` file of the Alfabet Server and of the clients.

1) Click the **Alfabet Aliases** node in the **Administrator** explorer.

2) In the table on the left, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** 🖉 button. An editor opens.

4) In the editor, go to the **Server Settings** tab and select the **Control Client Aliases** checkbox.

5) Click **OK** to save your changes.

Configuring User Authorization Based on Windows Sign-On Data or Data from Federated Authentication

Whether a Alfabet user is authorized to view or edit an object in the Alfabet database depends on the access permission concepts implemented in the Alfabet database. Some of the permission concepts depend on relations between the object class `Person` and access permission related to user group (class `UserGroup`) or user profile (class `ALFA_USERPROFILE`).

> (i) An overview of the access permission concepts for access to objects is provided in the section [Configuring User Authentication](#).

The assignment of users to user groups and user profiles can be maintained via the data from an external server such as an LDAP server used for Windows Sign-On or a SAML identity provider used for Federated Authentication. The relations defined in the external source can be used to update the `RELATIONS` table in the Alfabet database. During update, the external relations will either be added to the existing relations in the `RELATIONS` table or will substitute the defined user mapping to user profiles and user groups in the Alfabet database with the user mapping defined in the external source.

A detailed description of the required configuration is provided in the section [Configuring the Alfabet Web Application to Perform User Authentication Based On User Data from an External LDAP Data Source](#).

For federated authentication using SAML, authorization can optionally be performed by reading data directly from the SAML response instead of configuring the SAML identity provider as an external source via the **External Sources Configuration** functionality. The following configuration is required:

- [Configuring Custom Attributes for Storage of the Incoming Data for the Object Class Person](#)

- [Configuring the Mapping of Data and Updating Alfabet Internal Relations in the XML Object AuthConfiguration](#)

- [Activating User Authorization in the Remote Alias of the Alfabet Client](#)

Configuring Custom Attributes for Storage of the Incoming Data for the Object Class Person

The mechanism reading data from the SAML service response writes the data to properties of the object class `Person` for the person that is currently authenticated. Mapping data to the object class properties is defined in the next configuration step. User properties such as `Name` and `EMail` can be updated by writing data directly to the table of the object class `Person`. For user group assignment or user profile assignment, the object class property is of the class `ReferenceArray` and data is not stored in the object class `Person` but in the `Relations` table in the Alfabet database. This data must be stored in custom properties of the object class `Person` in a first import step. The data will be used to alter the `Relations` table in a second import step.

For each type of imported reference array data, a custom attribute of the **Type** `String` must be added to the object class `Person`. For detailed information about how to define a custom object class property, see the section *Configuring Custom Properties for Protected or Public Object Classes* in the chapter *Configuring the Class Model* of the reference manual *Configuring Alfabet with Alfabet Expand*.

The data is written in the attribute separated with a vertical bar:

```
Value1|Value2|Value3
```

Configuring the Mapping of Data and Updating Alfabet Internal Relations in the XML Object AuthConfig-uration

The XML object **_AuthConfiguration_** specifies the mapping of incoming data from the SAML response to the object class properties of the object class person as well as the update of the RELATIONS table of the Alfabet database with the data stored in the custom properties of the object class Person. The XML object can be edited in the tool Alfabet Expand or in the tool Alfabet Administrator. If you want to define different mappings for different Alfabet Clients, you can create copies of the XML object **_AuthConfiguration_** in the tool Alfabet Expand and define different mappings in each of the copies.

To edit the XML object **_AuthConfiguration_** in the tool Alfabet Expand:

1) Expand the **XML Objects** node in the explorer of the **Presentation** tab of Alfabet Expand.

2) Expand the **Administration** node.

3) Right-click **_AuthConfiguration_** and select **Edit XML** in the context menu. The XML object opens in the center pane.

4) Define the required XML elements in the editor.

5) If the XML object shall be visible and editable in the Alfabet Administrator, set the **Visible in Administrator** attribute in the attribute window of the XML object **_AuthConfiguration_** to True.

6) In the menu of Alfabet Expand , click the Save button to save your changes.

To create a copy of the XML object **_AuthConfiguration_** in the tool Alfabet Expand and define the configuration in the new XML object:

1) Expand the **XML Objects** node in the explorer of the **Presentation** tab of Alfabet Expand.

2) Expand the **Administration** node.

3) Right-click **_AuthConfiguration_** and select **New XML Object As Copy** in the context menu. A new XML object **_AuthConfiguration_1_** is added to the explorer.

4) Click the new XML object node in the explorer and optionally edit the following attributes in the attribute window:

   • **Name**: If applicable, edit the technical name of the XML object. The name must be unique and may not contain whitespaces or special characters.

   • **Visible In Administrator**: Set the attribute to True if you want the XML object to be visible and editable in the tool Alfabet Administrator.

5) Right-click the new XML object in the explorer and select **Edit XML** in the context menu. The XML object opens in the center pane.

6) Define the required XML elements in the editor.

7) In the menu of Alfabet Expand , click the Save button to save your changes.

To edit the XML object in the tool Alfabet Administrator:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer.

2) Right-click the server alias of the Alfabet Web Application for which you want to configure the XML object and select **Connect**.

3) In the explorer, click the **XML Objects** node.

4) In the table on the right side of the explorer, select ***AuthConfiguration*** or, if your configuration is based on a copy of the XML object ***AuthConfiguration***, the relevant copy, and click the

Edit ✏ button in the toolbar. An editor opens displaying the current configuration of the XML object.

> ⓘ If you do not see the XML object that you want to edit in the table, the **Visible in Administrator** attribute of the XML object is set to `False`. The attribute must be set to `True` using the tool Alfabet Expand to make the XML object visible in the Alfabet Administrator. Specify the attributes for XML objects in Alfabet Expand is explained above.

5) Define the required XML elements in the text editor.

6) Click **OK** to save your changes.

The XML object must consist of a root XML element `AuthConfiguration` containing two child elements:

- A child XML element `Actualization` that is a container for one or multiple XML elements `AuthAttrMapping`. For each object class property of the object class `Person` that shall be updated for the user login in via SAML, an XML element `AuthattrMapping` must be added to the XML object.

- A child XML element `Authorization` that is a container for one or multiple XML elements `AuthQuery`. For each relation that shall be updated on basis of the data in the `ALFA_EXTERNAL_RELATION` table, an XML element `AuthQuery` must be added to the XML object.

The following XML attributes must be defined for the XML element `AuthAttrMapping`:

XML Attribute of the XML Element AuthAttrMapping	Required Configuration
Name	The name of the SAML attribute.
Target Property	The Name of the object class property of the object class `Person` that the SAML attribute value targets.

The following attributes must be defined for the XML element ***AuthQuery***:

XML Attribute of the XML Element AuthQuery	Required Configuration
Mode	Set this XML attribute to one of the following values: • `Add`: Existing relations in the Alfabet`RELATIONS` table that does not exist in the `ALFA_EXTERNAL_RELATION` table are not deleted. If a relation exists in both tables, it is kept. If it exists in the `ALFA_EXTERNAL_RELATION` table only, it is added to the `RELATIONS` table.

XML Attribute of the XML Element AuthQuery	Required Configuration
	• `Replace`: All existing relations of the defined type are deleted from the Alfabet`RELATIONS` table prior to adding relations from the `ALFA_EXTERNAL_RELATION` table.
`TargetProp-erty`	Define the Name of the standard property of the object class `Person` that defines the relation to the target object class. Only relations defined via the specified property for the user currently logged in are included in the data update process.
`Query`	Define an Alfabet query or native SQL query that finds the target objects of the relation in the Alfabet database. In the `RELATIONS` table, relations are defined via three columns: • `FROMREF`: Defines the `REFSTR` of the object class referencing another object class. If a new relation is added to the `RELATIONS` table on basis of an ***AuthConfiguration*** definition, the `FROMREF` column is filled with the `REFSTR` of the user currently logging in. • `PROPERTY`: Defines the Name of the property of the `FROMREF` object class that stores the relation to the target object class. If a new relation is added to the `RELATIONS` table on basis of an ***AuthConfiguration*** definition, the `PROPERTY` column is filled with the value of the XML attribute `TargetProperty` of the relevant XML element `AuthQuery`. • `TOREF`: Defines the `REFSTR` of the target object for the relation. The query defined with the XML attribute `Query` of the XML element `AuthQuery` must return the `REFSTR` of the Alfabet object that is target of the relation. For each object found, a relation is added to the `RELATIONS` table. To find the relevant target objects in the Alfabet database, the mapping of the data stored in the relevant custom attribute of the object class `Person` to objects in the Alfabet database must be defined via the query.

The following example shows an XML object ***AuthConfiguration*** defining update of the assignment of a user to user profiles. The example is based on the following:

- The SAML attribute `memberOf` that contains the information about the user profile assignment is mapped to a custom property `ADBTO` of the object class `Person` in an XML element `AuthAttrMapping`.

- The property `PROFILES` of the Alfabet object class `Person` stores the relation to objects of the object class `ALFA_USERPROFILE`. This property is defined in the XML attribute `TargetProperty`.

- The parameter `CURRENT_USER` of the Alfabet query language is used in the queries to identify the user that currently logs in.

- The standard property `EXTERNAL_ID` of the Alfabet object class `ALFA_USERPROFILE` is used to define the mapping to the values stored in the custom property `ADBTO` of the object class `Person`.

- The query splits the string stored in the custom property `ADBTO` of the object class `Person` into the individual values that are then mapped against the standard property `EXTERNAL_ID` of the Alfabet object class `ALFA_USERPROFILE`.

```
<AuthConfiguration>

    <Actualization>

        ...

        <AuthAttrMapping Name="memberOf" TargetProperty="ADBTO"/>

    </Actualization>

    <Authorization>

        <AuthQuery Mode="Add" TargetProperty="PROFILES"

        Query="WITH TMP_USER(REFSTR, VALUE) AS (SELECT p.REFSTR,
        dbo.SCF_SplitString.value

            FROM PERSON P

            CROSS APPLY dbo.SCF_SplitString(P.ADBTO,'|')

            WHERE P.REFSTR = @CURRENT_USER)

        SELECT AUP.REFSTR FROM ALFA_USERPROFILE AUP

        JOIN TMP_USER ON TMP_USER.VALUE = AUP.EXTERNAL_ID"

        />

    </Authorization>

</AuthConfiguration>
```

ⓘ In Alfabet, the user profile assignment depends on the status of the user. If the status is anonymous, the user interface automatically opens with the viewer profiles marked as standard profile for anonymous user. If the status is `NamedUser`, user profiles must be assigned to the user. The user can then log in with any of the assigned user profiles. If no user profiles are assigned to a named user, the user cannot access the Alfabet user interface.

If user profiles are assigned to an anonymous user via the query definition in `AuthConfiguration`, the status of the user is automatically changed to `NamedUser`. But if the user profile definition for the user is removed in the external LDAP source and the update of the user profiles via `AuthConfiguration` leads to a detachment of all user profiles from the `NamedUser` in the Alfabet database, the user is not changed back to an anonymous user. If you want the user to access Alfabet with the viewer profile for anonymous access, you must define the query in the XML object ***AuthConfiguration*** to assign the view profile to all users or to all users for which no user profiles have been assigned in the external database.

Activating User Authorization in the Remote Alias of the Alfabet Client

To configure the remote alias to use authorization via relations stored in an external database:

1) Click the **Alfabet Aliases** node in the **Administrator** explorer. The right pane displays a list of all available alias configurations.

2) In the table, select the remote alias configuration that you want to edit.

3) In the toolbar, click the **Edit** 🖉 button. You will see the editor in which you can edit the remote alias configuration.

4) Go to the **ClientSettings** > **Authorization** tab to open it.

5) Enter the name of the XML object containing the authorization configuration in the **XML Object** field. The name is either `AuthConfiguration` or the name defined with the **Name** attribute of the copy of the XML object *AuthConfiguration*.

6) Click **OK** to save your changes.

Anonymizing Data

The data anonymization capability ensures data transparency and accountability across the enterprise as well as supports the enterprise to meet requirements of the General Data Protection Regulation (GDPR). By means of pseudonymization, for example, user data can be replaced with an artificial identifier to ensure anonymity if the user leaves the enterprise, or all sensitive data can be replaced with artificial identifiers in the development or test environment.

Anonymization can be performed for data of the type `String`, `Text`, `URL` and `Picture`. A solution designer must enable the anonymization function for the relevant object classes in Alfabet Expand to activate the anonymization options available for system administrators. The solution designer also configures individual anonymization methods for the object class properties that may be anonymized. Data can either be substituted with a random string, substituted with the `REFSTR` of the anonymized object, or set to `NULL` during anonymization.

ⓘ   For details about the configuration required to anonymize data and the consequences of anonymization, see *Anonymizing Data* in the reference manual *Configuring Alfabet with Alfabet Expand*.

If anonymization is triggered by one of the methods described below, the anonymization is applied to all data for an object class property if the object class is configured to be anonymized and an anonymization method is specified for the object class property. Anonymization is done for all objects of an object class. Anonymization can be performed for selected users only for user data.

Anonymization changes the following object class property values in the database:

- Values stored in the object class table of the object class in the Alfabet meta-model. If data translation is enabled, translated values are also anonymized.

- Values stored in the history table `<CLASSNAME>_AU` of the object class.

- Anonymization will be applied to the values in the columns `AUDIT_USER`, `CREATION_USER`, `LAST_UPDATE_USER` and `DELETE_USER` of all audit history database tables (`<CLASSNAME>_AU` and `RELATIONS_AU`), if one of the following applies:

    - If the object class property `USER_NAME` of the object class `PERSON` is anonymized, and the server alias for connection to the Alfabet database is configured to use the `USER_NAME` for auditing.

- If the object class property TECH_NAME of the object class PERSON is anonymized, and the server alias for connection to the Alfabet database is configured to use the TECH_NAME for auditing.

- If the object class property USER_NAME of the object class PERSON is anonymized with the method ToBeReplacedByKey, anonymization will also be applied to the **Last Update User** and **Creator** attributes in the **Tech Info** section of the configuration objects in Alfabet Expand. For all other anonymization methods these attributes are left unchanged.

  ⚠ If the anonymized user is the **Last Update User** or **Creator** of any configuration object subordinate to the **Classes** explorer node in the **Meta-Model** tab, the connections of all currently running Alfabet components with the Alfabet database will be terminated and the database will be locked during the anonymization process. The Alfabet components need to be restarted afterwards.

There are three ways to anonymize data:

- The **Anonymize Data** functionality anonymizes all values for all object class properties configured to be anonymized in the Alfabet meta-model of the current Alfabet database.

- The **Anonymize User Data** functionality anonymizes data for one or multiple selected users (for one or multiple selected objects of the object class PERSON) if the configuration of the object class PERSON specifies that the respective object class property shall be anonymized.

- The **Archive Current Database with Anonymized Data** functionality anonymizes all values for all object class properties configured to be anonymized in the Alfabet meta-model in a database archive file (ADBZ) during creation of the archive file. The data in the original Alfabet database that is archived is not affected.

In the Alfabet Administrator, all anonymization methods are available as options in the context menu of the server alias.

Alternatively, the console application AlfaAdministratorConsole.exe can be used to anonymize all object class property values configured to be anonymized in an Alfabet database or data for one or multiple selected users.

The following information is available:

- Anonymizing All Relevant Data in the Alfabet database

  - Anonymizing All Relevant Data Using the Alfabet Administrator

  - Anonymizing All Relevant Data via a Command Line Tool

- Anonymizing User Data

  - Anonymizing Data of Selected Users with the Alfabet Administrator

  - Anonymizing Data via a Command Line Tool

  - Excluding Users from Anonymization

- Creating a Database Archive File Containing Anonymized Data

Anonymizing All Relevant Data in the Alfabet database

The **Anonymize Data** functionality anonymizes all values for all object class properties configured to be anonymized in the Alfabet meta-model of the current Alfabet database.

To anonymize data, you can either use the Alfabet Administrator or the command line tool `AlfaAdminis-tratorConsole.exe`:

- [Anonymizing All Relevant Data Using the Alfabet Administrator](#)

- [Anonymizing All Relevant Data via a Command Line Tool](#)

Anonymizing All Relevant Data Using the Alfabet Administrator

To trigger anonymization of data in the Alfabet Administrator:

> ⚠️ Anonymizing data is a sensible process that might disrupt database integrity. It cannot be reverted! **Always back up the Alfabet database prior to triggering data anonymization!**

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias connecting to the Alfabet database that you want to archive.

2) Select **Anonymize Data**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) In the **Anonymize Data** window that opens, define the location for the log file that shall be used to log information about the anonymization process. If you select an already existing log file, the standard message of the selector window informs you that the file will be overwritten. Nevertheless, the log information is appended to the existing content of the selected file.

5) Click **View Content Summary** and check whether the current anonymization configuration in your Alfabet database is as expected and then close the summary window. The following information is displayed:

```
Number of classes to be anonymized: 6
  Anonymize User Info in Audit Tables: False

Class: Application
 - Property: ShortName => ToBeNullified
 - Property: Version => ToBeLeftUnchanged
 - Property: SC_Sox_RelevantDescription => ToBeRandomized
 - Property: SC_RM_Comment => ToBeRandomized
 - Property: SC_DM_UpdateDescription => ToBeRandomized
 - Property: SC_DistributionBasis => ToBeLeftUnchanged
 - Property: SC_SecurityClarification => ToBeLeftUnchanged
 - Property: ID => ToBeRandomized
 - Property: Name => ToBeReplacedByKey
 - Property: Description => ToBeRandomized
 - Property: SC_VersionID => ToBeLeftUnchanged
```

- **Number of classes to be anonymized**: The overall number of object classes for which the **Anonymize** attribute is set to `True`.

- **Anonymize User Info in Audit Tables**: Informs about the anonymization in audit history tables (`<CLASSNAME>_AU` and `RELATIONS_AU`) in the Alfabet database and in the **Creator**

and **Last Update User** attributes in the **Tech Info** section of the configuration objects in Alfabet Expand. The **Anonymize** attribute must be set to `True` for the object class `Person` and a method other than `ToBeLeftUnchanged` must be selected for the object class property `USER_NAME` and/or for the object class property `TECH_NAME` of the object class `Person`.

- `True`: The user name will then be anonymized in all audit history tables and in the attributes **Creator** and **Last Update User** in the **Tech Info** section of the configuration objects in Alfabet Expand.

- `True (Technical Info Not Anonymized)`: The user name will only be anonymized in the audit history tables. The **Creator** and **Last Update User** attributes in the **Tech Info** section of the configuration objects in Alfabet Expand is not anonymized. This means that the `TECH_NAME` is anonymized whereas the `USER_NAME` is left unchanged. The server alias configuration specifies that the `TECH_NAME` is written to the audit history.

- `False (Technical Info Anonymized)`: The user name will only be anonymized in the attributes **Creator** and **Last Update User** in the **Tech Info** section of the configuration objects in Alfabet Expand. The user information in the audit history tables is not anonymized. This means that the `USER_NAME` is anonymized, whereas the `TECH_NAME` is left unchanged. The server alias configuration specifies that the `TECH_NAME` is written to the audit history.

- `False`: Audit history tables and the **Creator** and **Last Update User** attributes in the **Tech Info** section of the configuration objects in Alfabet Expand are not anonymized.

- **Class**: For each object class for which data will be anonymized, all object class properties that may be subject to anonymization are listed with information about the configured anonymization method.

6) Click **Anonymize**.

Anonymizing All Relevant Data via a Command Line Tool

You can use a command line tool provided by Software AG to trigger the anonymization of data.

⚠ Anonymizing data is a sensible process that might disrupt database integrity. This cannot be re-verted! **Always back up the Alfabet database prior to triggering data anonymization!**

Executable	`AlfaAdministratorConsole.exe` located in the **_Programs_** subdirectory of the Alfabet installation directory
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see [Standard Logging for Alfabet Batch Utilities](#).
Command line help	Start executable with `-h` or `-help`

The executable must be started with the following parameters:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <user password>-db_anonymize_mm
```

The table below displays the command line options:

Command Line Option	Manda-tory/ Default	Explanation
`-db_anonymize_mm`	Manda-tory	To anonymize all data configured to be anonymized in an Alfabet database, start the console application with `-db_anonymize_mm`
`-msalias <alias name>`	Manda-tory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.
`-msaliasesfile <Alfabet configuration file path>`	Optional	If the `AlfabetMS.xml` configuration file that contains the specifi-cation of the alias is not located in the same directory as the exe-cutable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName <Alfabet user name>`	Manda-tory	Alfabet user name for login. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPass-word <user password>`	Optional	Alfabet login password.

Anonymizing User Data

The **Anonymize User Data** functionality is available to anonymize data for one or multiple selected users (for one or multiple selected objects of the object class `Person`).

Data for a selected user is only anonymized if anonymization is enabled in the configuration of the object class `Person` and if the user is not explicitly excluded from anonymization.

To anonymize user data, you can either use the Alfabet Administrator or the command line tool `AlfaAdministratorConsole.exe`:

The following information is available:

- [Anonymizing Data of Selected Users with the Alfabet Administrator](#)

- [Anonymizing Data via a Command Line Tool](#)

- [Excluding Users from Anonymization](#)

Anonymizing Data of Selected Users with the Alfabet Administrator

To trigger anonymization of the data of selected users:

> ⚠ Anonymizing data is a sensible process that might disrupt database integrity. This cannot be reverted! **Always back up the Alfabet database prior to triggering data anonymization!**

> ⚠ If the anonymized user is the **Last Update User** or **Creator** of any configuration object subordinate to the **Classes** explorer node in the **Meta-Model** tab, the connections of all currently running Alfabet components with the Alfabet database will be terminated and the database will be locked during the anonymization process. The Alfabet components need to be restarted afterwards.

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias connecting to the Alfabet database that you want to archive.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) Right click the server alias again and select **Anonymize User Data**.

5) In the window that opens, select one or multiple users to be anonymized in the table. You can set the following filters and click **Update** to search for specific users:

   • **Search Pattern**: Enter a search pattern to search for in either all standard attributes or in the attribute selected in the drop-down field of the field on the right of the **Search Pattern** field.

   • **Profile**: Select a user profile in the drop-down field to limit the display in the table to users assigned to the selected user profile.

   • **Show Alfabet-Managed Users Without Password**: Select the checkbox to limit the display in the table to users that are logging in with a user name and password managed in Alfabet and that currently have no password assigned.

6) In the **Log File** field, define the location for the log file that shall be used to log information about the anonymization process. If you select an already existing log file, the standard message of the selector window will explain that the file will be overwritten. Nevertheless the log information is appended to the existing content of the selected file.

7) Click **Anonymize**.

Anonymizing Data via a Command Line Tool

To trigger anonymization of data, you can use a command line tool provided by Software AG.

> ⚠ Anonymizing data is a sensible process that might disrupt database integrity. This cannot be reverted! **Always back up the Alfabet database prior to triggering data anonymization!**

Executable   `AlfaAdministratorConsole.exe` located in the ***Programs*** subdirectory of the Alfabet installation directory

Logging	Standard Alfabet logging. For information about standard logging and the command line options, see [Standard Logging for Alfabet Batch Utilities]().
Command line help	Start executable with`-h` or`-help`

The executable must be started with the following parameters:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <user password>-db_anonymize_user-userRefs
<PersonREFSTR,PersonREFSTR>
```

The table below displays the command line options:

Command Line Option	Mandatory/ Default	Explanation
`-db_anony-mize_user`	Mandatory	To anonymize user data for selected users in the Alfabet database, start the console application with `-db_anonymize_user`
`-userRefs <PersonREFSTR,Person-REFSTR>`	Mandatory	Specify the value of the property `REFSTR` of the user that shall be anonymized. User data is stored in the object class `Person`. If the data of multiple users shall be anonymized, the `REFSTR` values can be listed comma-separated. The `REFSTR` property is not displayed on standard views in Alfabet. To see the REFSTR of a user, you can define a configured report in Alfabet Expand or read the information via RESTful service requests.
`-msalias <alias name>`	Mandatory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.
`-msaliasesfile <Alfabet configuration file path>`	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName <Alfabet user name>`	Mandatory	Alfabet user name for login. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.

Command Line Option	Manda-tory/ Default	Explanation
`-alfaLoginPass-word` <user pass-word>	Optional	Alfabet login password.

Excluding Users from Anonymization

Single users can be excluded from anonymization to make sure that the data of that user cannot be anonymized:

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias connecting to the Alfabet database that you want to archive.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) Expand the server alias node and click the **User Management** > **Users** node. A workspace for user management opens.

5) In the table, select the user that you want to exclude from anonymization.

6) In the toolbar, click **Edit** . The **User** editor opens.

7) Open the **Permissions** tab and select the **Excluded From Anonymization** checkbox.

8) Click **OK** to save your changes.

Creating a Database Archive File Containing Anonymized Data

The **Archive Current Database with Anonymized Data** functionality is available to anonymize all values for all object class properties configured to be anonymized in the Alfabet meta-model in a database archive file (ADBZ) during creation of the archive file. The data in the original Alfabet database that is archived is not affected. This method is useful to archive the content of a productive database in an anonymized form to implement it in a test or development environment without information about sensitive data.

To archive the non-anonymized data in the current database and apply anonymization to the data in the archive file:

1) In the explorer of the Alfabet Administrator, expand the **Alfabet Aliases** node and right-click the server alias connecting to the Alfabet database that you want to archive.

2) Select **Connect**. A login window is displayed.

3) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

4) Right click the server alias again and select **Archive Current Database with Anonymized Data**. The **Alfabet Archive Manager (Anonymized Data)** opens.

5) In the window that opens, click the **Browse** button on the right of the **The file in which you want to archive the Alfabet database** field to define the location for the database archive file on the local file system. The default name of database archive file contains the string `Anonymized` to distinguish between anonymized and non-anonymized archives that shall be used to log information about the anonymization process.

6) Select the **Squeeze Audit Tables** checkbox if you would like the audit tables to be cleaned prior to archiving the database. Any entries that were generated during a batch update of data via batch utilities without documenting any audit relevant changes, for example, will be deleted from the audit tables prior to archiving the database.

7) Select the **Include User Settings** checkbox if you want user settings defined for the current database to be added to the archive.

8) Click **Archive**.

To overwrite the content of a database with the data from an anonymized database archive file, use the **Restore Database Archive** option in the context menu of the server alias. For more information, see Using the Context Menu Available via the Alias Explorer Node.

⚠ Anonymizing data is a sensible process that might disrupt database integrity. It cannot be reverted! **Always back up the Alfabet database prior to restoring it with an anonymized archive file!**

Chapter 6: Configuring and Activating Alfabet Functionalities

After installing the Alfabet components, the solution environment needs to be configured and maintained to meet the specific customer demands. Most functionalities are delivered ready to use in a default configuration, but others may require any of the following configurations:

- Activation of the feature or a set of configuration steps that must be carried out in the configuration tool Alfabet Expand.

- Routine execution of batch jobs at regular intervals. This is required for a few specific functionalities. Batch jobs can be optionally executed for a number of other functionalities in order to enhance usability in data processing.

The reference manual *System Administration* describes the configuration steps that are carried out in the tool Alfabet Administrator as well as the execution of batch jobs. Any configuration required in the configuration tool Alfabet Expand will reference the relevant section in the reference manual *Configuring Alfabet with Alfabet Expand*.

The following information is available:

- Overview of Alfabet Functionalities Requiring Batch Job Processing and/or Activation

- Configuring the Graphic Display of the Alfabet Interface

  - Enabling the Display of Images and Videos from External Web Servers

  - Allowing Users to Change the Color Assigned to Objects in Business Graphics

  - Activating Roboto as the Standard Font for the Alfabet Interface

- Making Documents and Files Available to the Alfabet User Community

  - Uploading Documents via User Interaction to the Internal Document Selector

  - Uploading Documents to a Default Folder in the Internal Document Selector

  - Uploading Documents to a Folder on the Local File System

- Activating the Dispatch of Email Notifications in Alfabet

  - Configuring the Alfabet Server to Connect to the SMTP Server

    - Specifying the SMTP Server in the Server Alias of the Alfabet Server

    - Testing the Connection to the SMTP Server

  - Specifying Sender Email Addresses

  - Re-Routing Emails to a Defined Address for Testing

  - Configuring the Setup To Open the Alfabet Interface via Links in Email Notifications

    - Access Permissions To Open the Alfabet User Interface

    - Defining the User Profile Used To Open the View

    - Permissions to Edit Objects

    - Navigation Options for Users Accessing Alfabet via a Hyperlink

Overview of Alfabet Functionalities Requiring Batch Job Processing and/or Activation

This section provides an overview of the functionalities in Alfabet that require configuration by means of the configuration tool Alfabet Expand, the tool Alfabet Administrator, and/or the configuration and execution of batch jobs.

> The text displayed in automatically-generated email notifications is also configured in Alfabet Expand. For more information, see the section *Configuring Text Templates for Email Notifications* in the reference manual *Configuring Alfabet with Alfabet Expand*.

Alfabet Function-ality	Tool for Configuration of Functionality	Batch Job Required	Configu-ration of Email Dispatch
Upload of a com-pany logo for dis-play in the Alfabet user interface	Alfabet Expand	No	No
Customization of the coloring of ob-jects in business graphics	Alfabet Administrator and Alfabet Expand	No	No
Sending emails with links to Alfabet views	Alfabet Administrator	No	Yes
Storage of attach-ments in the local file system instead of in the Alfabet database	Alfabet Administrator	No	No
Assignments	Alfabet Expand	To dispatch reminder emails	Yes
Assignments for business process and organizational changes	Alfabet Expand	To automatically generate assignments and email dispatch	Yes
Monitors	Alfabet Expand and **Monitors** functionality in the Alfabet user in-terface	To dispatch emails	Yes
Workflows	Alfabet Expand and **Workflow Administra-tion** functionality in the Alfabet user interface	Depending on the workflow configuration, to automatically start workflows, trigger workflow steps, and dispatch emails re-garding delegated, escalated, finished, paused, refused, and resumed workflow steps, change of workflow step owner, or overdue workflow steps	Yes

Alfabet Function-ality	Tool for Configuration of Functionality	Batch Job Required	Configu-ration of Email Dispatch
Discussion Groups	**Discussion Groups** functionality in the Alfabet user interface	No	Yes
Evaluations	**Evaluations and Port-folios** functionality in the Alfabet user inter-face	To compute indicators	No
Color Rules for Map Views	**Color Rules** page view for a map view in the Alfabet user interface	To execute color rules based on Alfabet queries	No
Auditing the Change History of Objects	Alfabet Expand and Alfabet Administrator	To initiate the audit tables.	No
Full-Text search	Alfabet Expand **Full-Text Search** function-ality in the Alfabet user interface	To initiate and update the index for the full-text search	No
Reports	Alfabet Expand	To delete expired reports	No
Publications	Alfabet Expand	No	No
Form-Based Data Collection	Alfabet Expand	To upload forms and recollect them	No
Value Manage-ment	Alfabet Expand	No	No
Project Planning	Alfabet Expand	No	No
Compliance Man-agement	Alfabet Expand	No	No

Alfabet Function-ality	Tool for Configuration of Functionality	Batch Job Required	Configu-ration of Email Dispatch
Domain Manage-ment	Alfabet Expand	No	No
Collaboration	Alfabet Administrator (The functionality must be activated in the server alias of the Alfabet Web Applica-tion)		

Configuring the Graphic Display of the Alfabet Interface

The design of the standard Alfabet interface contains customizable elements that allow the customer to adapt the solution interface to the company's corporate design. The following options are available:

- A company logo can be displayed on the Alfabet interface.

- Custom icons can be uploaded that can be used to visualize object classes and object class stereotypes, matrix objects, indicator values, etc.

- The layout of the Alfabet interface can be adapted to the customer's corporate design by changing colors and font types.

- Navigation pages can be designed and uploaded to the Alfabet database. A navigation page serves as the start page for a user profile. A navigation page is a customized HTML file with hyperlinks that provide Alfabet users access to the software and support them in their work. A navigation page could include, for example, informational text or images about enterprise-specific workflows in the Alfabet solution, links to specific functionalities in Alfabet, URL links to internal documents or the Web, or hyperlinks to other navigation pages.

- Alfabet users may be provided with authorization to change the color used to identify specific objects in business graphics.

Customization of the user interface is typically done by a solution designer using the tool Alfabet Expand and the web-based Guide Pages Designer that is available as part of Alfabet Expand to configure navigation pages. But some of the customizations are triggered by configuration of the server alias of the Alfabet Web Application or require configuration of other involved components that are typically done by a system administrator. This section informs about the features that require system administrator tasks:

- [Enabling the Display of Images and Videos from External Web Servers](#)

- [Allowing Users to Change the Color Assigned to Objects in Business Graphics](#)

- [Activating Roboto as the Standard Font for the Alfabet Interface](#)

Enabling the Display of Images and Videos from External Web Servers

The `Content-Security-Policy` settings in the web.config of the Alfabet Web Application disable the display of images or media from other web servers. If you would like to display content from other web servers in the Alfabet user interface such as videos embedded in automated assistants or images embedded in HTML text, you must extend the `Content-Security-Policy` definition with the specification of the image or media source. For more information, see the documentation of the `Content-Security-Policy` HTTP header at, for example, https://content-security-policy.com/.

Allowing Users to Change the Color Assigned to Objects in Business Graphics

The color used to display all objects in an object class can be specified in the class settings of the object class. However, users may also explicitly specify a color for objects in object classes that have a ***Color***

property and that the user has access permissions to. This is carried out via the **Colors** button which is available in standard Alfabet views and configured reports with business graphics (for example, portfolios and bar charts) in the Alfabet user interface. Please note that if the user changes the color of an object, the selected color is used for the object in all views and reports for all Alfabet users. This capability is typi-

cally used for demonstration purposes, and it is highly recommended that the **Colors** button is hidden from Alfabet views and configured reports in order to prevent users from changing object colors across the user community. The server setting **Allow Edit Object Colors** should therefore be specified to hide the

Colors button in all Alfabet views and configured reports.

> Object colors may also be defined via a color selector configured in a custom editor created for an object class that has a ***Color*** property. When the user creates or edits an object in the custom editor, he/she can change the color in the custom editor. The color will then be used for the object in all views and reports for all Alfabet users.
>
> Please note that the server setting **Allow Edit Object Colors** explained below is not relevant for the color selector configured for a custom editor. If a color selector is configured for a custom editor, it will be implicitly available in the custom editor for users having access permissions to the custom editor. For more information about configuring the color selector for a custom editor, see the section *Adding a Color Selector to the Custom Editor* in the chapter *Configuring Custom Editors* in the reference manual *Configuring Alfabet with Alfabet Expand*.

If you do not want users to change the color assigned to objects in views and reports, you can disable the

Colors button with one of the following mechanisms:

- To remove the **Colors** button from the toolbar of views and reports for all users, you can disable the functionality in the configuration of the server alias of the Alfabet Web Application. The required setting is described below.

- To remove the **Colors** button from the toolbar of reports for a set of users, you can disable the button per view scheme. For more information, see *Configuring a View Scheme for a User Profile* in the reference manual *Configuring Alfabet with Alfabet Expand*.

To configure the Alfabet Web Application to hide the **Colors** button in Alfabet views and configured reports:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer in the Alfabet Administrator.

2) In the table, select the alias that you want to edit and click the **Edit** ✏️ button in the toolbar. An editor opens.

3) In the **Server Settings** tab, deselect the checkbox for **Allow Edit Object Colors**.

4) Click **OK** to save your changes.

Activating Roboto as the Standard Font for the Alfabet Interface

GUI schemes can be configured in Alfabet Expand to align the Alfabet user interface with your corporate design. If no customer configuration is available, the GUI scheme labelled `Default Scheme` is used for the display of the Alfabet user interface. The protected GUI scheme `SoftwareAG_101Styles` is available for reference purposes and specifies all standard settings in the Alfabet user interface.

The **Application Font** attribute of the default GUI scheme is set to `Roboto`. The `Roboto` font is not a standard font available via the web browser, and therefore must be uploaded from the Alfabet Web Application at runtime. The upload of runtime fonts is not permitted per default in the configuration of the Alfabet Web Application. Therefore, if the `Roboto` font should be used for the user interface, the use of runtime fonts must be explicitly enabled via the server alias.

If the `Roboto` font is not explicitly enabled via the server alias, the fonts defined for the **Application Font Fallback 1** and **Application Font Fallback 2** attributes of the GUI scheme will be used in the Alfabet user interface.

To enable the display of `Roboto` font in the Alfabet user interface:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer in the Alfabet Administrator.

2) In the table, select the server alias of the Alfabet Web Application and click the **Edit** ✏️ button in the toolbar. An editor opens.

3) In the **Server Settings** tab, select the **Allow Usage of Run-Time Fonts** checkbox.

4) Click **OK** to save your changes.

Making Documents and Files Available to the Alfabet User Community

The **Internal Document Selector** is a central repository in Alfabet that allows users to access documents within the Alfabet system. For example, for many object classes in the Alfabet meta-model, Alfabet users can attach documents to individual objects in the **Attachments** page view available in the object's profile. A document attached to an object can be downloaded by users with access permission to the object in the **Attachments** page view of the object. Furthermore, the **Internal Document Selector** allows system-specific files to be uploaded in order to make them available to the Alfabet user community or the Alfabet solution. For more information about managing files in the **Internal Document Selector**, see the section *Uploading Documents and Managing User Permissions to Document Folders in the Internal Document Selector* in the reference manual *User and Solution Administration*.

ⓘ ZIP files are checked during upload for the size of content after decompression. The file will not be uploaded if the decompressed size is more than 100 percent of the compressed size or if storing the file content on the local drive would result in less than 1 GByte free space remaining or if any deviations from normal compression mechanisms are detected.

There are three methods to manage files assigned as attachment to Alfabet objects either in the **Internal Document Selector** or on the local file system:

- File storage in the **Internal Document Selector** without direct user access to the **Internal Document Selector**

- File storage in the **Internal Document Selector** with direct user access to folders and documents in the **Internal Document Selector**

- File storage on the local file system without direct user access to the storage location

ⓘ For security reasons, a blacklist and whitelist concept has been introduced to restrict uploading and downloading files with permissible file extensions in Alfabet. This is relevant for files uploaded or downloaded via the **Internal Document Selector** or stored in an external file system. An error message will be displayed if a user attempts to upload or download an impermissible file extension type. For more information about restricting file extensions, see the section *Specifying the Permissible File Extensions for Uploading/Downloading Files* in the reference manual *Configuring Alfabet with Alfabet Expand*.

Anti-virus scans will be performed for documents during upload to the **Internal Document Selector** , if the `scan_malware` key in the `Alfabet.config` file is set to `true`:

```
<add key="scan_malware" value="true"/>
```

The scan will be performed with the anti-virus scanner implemented on the server hosting the Alfabet Web Application. If no anti-virus scanner is explicitly implemented, the anti-virus capabilities of the underlying Windows® operating system will be used to scan document during upload.

Uploading Documents via User Interaction to the Internal Document Selector

If the document storage type `IDoc` is selected for the Alfabet Web Application, attachments will be uploaded to the Alfabet database and stored in the **Internal Document Selector**.

When the user selects **New** > **Upload Document** in the **Attachments** page view, the **Internal Document Selector** opens.

Depending on access permissions, the user can

- only attach files that are already uploaded to the **Internal Document Selector**,

- or he/she can also manage files in the **Internal Document Selector**. This includes uploading documents from the local file system to the **Internal Document Selector**, updating or deleting documents, and creating new folders.

The files can be stored in a folder hierarchy. The name of the document in the **Internal Document Selector** is the name of the document before upload.

Uploading a file to the **Internal Document Selector** and attaching a file to an object in the **Attachments** page view are two distinct processes. A single file in the **Internal Document Selector** can be attached to

multiple objects. In the **Attachments** page view, files can be detached, which will remove the link to the document in the **Internal Document Selector**. Nevertheless, the document will still be stored in the **Internal Document Selector** and can be assigned to other objects as attachment. If the same file is attached to multiple objects via the **Attachments** page view, the attachments can all refer to the same file in the **Internal Document Selector**.

Files can only be deleted directly in the **Internal Document Selector**. They will then be removed from the **Attachments** page view of any object they were assigned to.

The **Internal Documents** functionality of Alfabet allows the administrator to configure access permissions for folders in the **Internal Document Selector**. For detailed information about how to configure access permissions for the **Internal Document Selector**, see the section *Uploading Documents and Managing User Permissions to Document Folders in the Internal Document Selector* in the reference manual *User and Solution Administration*.

Each Alfabet user with the relevant access permissions can create sub-folders or delete and update files.

The `IDoc` mode is the default mode for attachment handling in Alfabet. for a new server alias configuration, the parameter **Document Storage Type** in the **Server Settings** > **General** tab will be set to `IDoc` by default. No further configuration is required.

Uploading Documents to a Default Folder in the Internal Document Selector

If the document storage type `DefaultIDocFolder` is selected for the Alfabet Web Application, attachments will be uploaded to the Alfabet database and stored in the **Internal Document Selector**.

When the user selects **New** > **Upload Document** in the **Attachments** page view, a standard browser opens. The document selected by the user using the standard browser is uploaded to the defined default directory in the **Internal Document Selector**. the user has no access to the Internal Document Selector via the **Attachments** page view.

The name of the document in the **Internal Document Selector** is the name of the document before upload extended with a GUID. The GUID is not displayed to the user on the Attachments page view.

If the user attaches the same file to another object, the file is stored with another GUID. There is one file for each attachment in the default folder.

If a user deletes a file in the **Attachments** page view, the file is deleted from the **Internal Document Selector**.

To update a file, the file must be downloaded from the **Attachments** page view, changed, and uploaded as a new attachment. The old version can either be kept in parallel or deleted via the **Attachments** page view.

Only administrative users can access the **Internal Document Selector** in the **Internal Documents** functionality of Alfabet. The administrator can configure access permissions for folders in the **Internal Document Selector** for other functionalities. For the default folder, all users can upload, download, and delete attachments independent of the access permissions defined by the administrator. If the administrative user deletes a file from the default folder, the file will no longer be available via the **Attachments** page view of the object it was assigned to.

To implement attachment upload without user access to the **Internal Document Selector**:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer in the Alfabet Administrator.

2) In the table, select the server alias that you want to edit and click the **Edit** ✎ button in the toolbar. An editor opens.

3) In the **Server Settings** tab, set the following fields:

- **Document Storage Type**: Select `DefaultIDocFolder` in the drop-down field.

- **Document Storage Location**: Specify the name of the folder in the **Internal Document Selector** for storage of attachments. The folder must be available under the root level of the **Internal Document Selector**. The folder will be created if a folder with the defined name does not exist.

4) Click **OK** to save your changes.

Uploading Documents to a Folder on the Local File System

If the document storage type `ExternalFileSystem` is selected for the Alfabet Web Application, attachments will be uploaded to the defined local upload directory. They are not stored in the Alfabet database. The database only contains information about the file name and location.

All files are stored directly in the specified upload directory. Subdirectories cannot be created. The file name consists of the original file name before upload and an automatically generated number. If the same file is uploaded multiple times (for example, because it is attached to multiple objects or because it is downloaded, changed, and uploaded again), the same file will be stored multiple times with the file name differing in the assigned number.

The local upload directory must be managed by an administrator with relevant access permissions on the local file system. The Alfabet Web Application writes a new file to the upload directory when the file is attached to an object by a user, but it is not deleting files automatically when they are detached from an object.

The local upload directory is not protected by Alfabet access permissions but by the access permissions granted on the local network for the directory. The Alfabet Web Client must have read/write access permissions to the directory. In the context of the Alfabet user interface, each user that has access to an object can upload and download documents and detach them from the object. Users do not directly access the download location but request files from the Alfabet Web Application. The upload location is not visible to the user.

Please note the following:

- The following configuration only changes the storage behavior for files uploaded by Alfabet users on the **Attachments** page view. All other files uploaded to the Alfabet database in the context of other functionalities are still uploaded to different subfolders of the **Internal Document Selector** and stored in the database.

- It is generally recommended that you do not store external links on the file system.

- If you change from using the **Internal Document Selector** to using a local upload directory for the files attached to objects and the documents have already been uploaded to the **Internal Document Selector**, the files uploaded to the **Internal Document Selector** will not be stored on the local upload directory but will remain in the Alfabet database. Nevertheless, they will remain visible in the **Attachments** page view and the user can download the document, if needed. It is not evident to the user which document is stored in which location.

- If you change from using a local upload directory to using the **Internal Document Selector** for the files attached to objects and the documents have already been stored in the local upload directory, the files stored in the local upload directory will remain visible in the **Attachments** page view but will no longer be accessible.

The **Internal Document Selector** is used per default for the upload of files. To specify that files should be uploaded via an external file system instead:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer in the Alfabet Administrator.

2) In the table, select the server alias that you want to edit and click the **Edit** ✎ button in the toolbar. An editor opens.

3) In the **Server Settings** tab, set the following fields:

   - **Document Storage Type**: Select `ExternalFileSystem` in the drop-down field.

   - **Document Storage Location**: Specify the absolute path to the location in the local file system where documents uploaded to the Alfabet interface are stored. Click the ***Browse*** button to select a directory.

     > ⓘ The Alfabet Web Application must have read/write access permissions to the selected directory.
     >
     > Server variables can be used to define part of the information as a variable with the value set in the **Variables** tab of the server alias configuration. For more information, see Defining Connections Based On Server Variables.

4) Click **OK** to save your changes.

5) Configure the Web browsers on the client hosts to allow upload of files from the local file system. The following extensions must be downloaded for the different browser types:

   - Google Chrome®: Install the extension Local Explorer- File Manager available for the browser.

   - Mozilla® Firefox®: Install the extension LocalLink available for the browser. Please note that files can only be opened via right-clicking the link and selecting the option "Open link in local context.

   - Microsoft® Edge®: Currently no additional settings are required.

Activating the Dispatch of Email Notifications in Alfabet

There are a number of functionalities in Alfabet that feature the automatic dispatch and delivery of emails from the Alfabet Web Application to Alfabet users. These functionalities require the configuration of the Alfabet components to send the emails and, in some cases, the regular execution of batch jobs to actually dispatch emails.

Emails are sent asynchronously via one or multiple Alfabet Server s connected to the same Alfabet database the Alfabet Web Application is connected to. This method ensures that emails are sent out even if a high number of users simultaneously trigger emails being sent and even if the user session terminates prior to completing the sending of the emails.

When a user triggers sending of an email via an action on the Alfabet user interface, the Alfabet Web Application writes an entry with information about sender, recipient and content of the email and the status `Pending` into the database table `ALFA_EMAIL_BUS`. The Alfabet Server scans the database table `ALFA_EMAIL_BUS` regularly for new email entries with the status `Pending` and sends the emails out via an external SMTP server. The status of the email is set to `Executing` as soon as the Alfabet Server starts sending the email. The status of email messages that have been processed by the Alfabet Server informs about the success of the sending process. If sending of an email fails, an error message is written into the `ALFA_EMAIL_BUS` table.

On the Alfabet user interface, administrators can check the success of the sending of emails in the **Email Message Log** (`ADMIN_MessageLogging`) functionality.

> (i) For information about the **Email Message Log** functionality, see the chapter *Tracking the Email Messages Sent in the Context of Alfabet Functionalities* in the reference manual *User and Solution Administration*.

All Alfabet functionalities for which the email capability is to be implemented require the setup of a connection to an SMTP server for outgoing emails. The following describes the general setup of the connection:

- Configure the Alfabet Server to send out emails. A connection to an SMTP server must be configured. For more information, see the section Configuring the Alfabet Server to Connect to the SMTP Server.

- Make sure that an email address is specified for each Alfabet user in the user configuration. For more information, see the section Specifying Sender Email Addresses.

- You should also consider defining parameters to specify access conditions to Alfabet views when opened from a hyperlink in an email. For more information about the configuration and impact of the parameters, see the section Configuring the Setup To Open the Alfabet Interface via Links in Email Notifications.

> Additional requirements that are relevant to emails sent out in the context of specific Alfabet functionalities are described in the following sections:
>
> - Configuring the Express View (Email) Capability
>
> - Triggering Target Date Control for the Assignments Capability
>
> - Batch Processes for Workflows with AlfaWorkflowCommandPrompt.exe
>
> - Batch Processing for Monitors and Change Management with AlfaBatchExecutor.exe

Configuring the Alfabet Server to Connect to the SMTP Server

Do the following to configure the sending of emails via SMTP server:

- [Specifying the SMTP Server in the Server Alias of the Alfabet Server](#)

- [Testing the Connection to the SMTP Server](#)

Specifying the SMTP Server in the Server Alias of the Alfabet Server

To send emails in the context of Alfabet functionalities, either a local SMTP server must be installed and configured on the Alfabet Server machine, or an existing enterprise SMTP server must be configured in the configuration file of the Alfabet Server.

The sender email address of Alfabet emails is configurable. The configuration is described in the following section. The sender email address must be a valid email address that must be available before you begin to configure the Alfabet Server

Configuration of the Alfabet Server comprises both the specification of the SMTP server and the parameters for processing emails in the queue.

The Alfabet Server checks the queue for pending emails in a configurable time interval and sends out emails to the SMTP server in blocks via a single connection. For the email queue processing parameters, best practice default values are displayed in the respective fields in a new server alias configuration. These parameters should only be changed if problems are encountered with the connection to the SMTP server.

To configure the Alfabet Server to connect to the SMTP server for outgoing emails:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer.

2) In the table, select the server alias that you want to edit and click the **Edit** ✎ button in the toolbar. An editor opens.

3) In the **Server Settings** > **Email Settings** tab, specify the connection to the SMTP server with the following parameters:

   - **EMail Mode**: This field is set to `SMTP` and for informational purposes only.

   - **SMTP Settings** section:

      - **Host**: Enter the fully qualified domain name of the SMTP server.

         > ⓘ By default, the Alfabet Server tries to use a local SMTP server with default settings for the connection definition. Therefore, the configuration of the **SMTP Settings** attributes is not required if you want the Alfabet Server to connect to an SMTP server that is installed on the Alfabet Server host on port 25 without authentication and without using SSL.

      - **Port**: Enter the port of the SMTP server for incoming connections. By default, port 25 is used.

      - **Use SSL**: Select the checkbox to send data to the SMTP server on an SSL secured connection. By default SSL is not used.

- **Enable Authentication**: Select the checkbox if authentication is required for the SMTP server. If the checkbox is selected, you must provide a user name and password in the **User** and **Password** attributes. By default, authentication is not enabled.

- **User**: If the **Enable Authentication** attribute is activated, enter the user name for authentication at the SMTP server.

- **Password**: If the **Enable Authentication** attribute is activated, enter the password for authentication at the SMTP server. The password is stored encrypted in the `AlfabetMS.xml` server alias configuration file.

- **Email Queue Sleep Time (milliseconds)**: The Alfabet Server will check the queue for emails to be sent out in the specified time interval. The default value is 3000.

- **Max. Emails per Minute**: Enter the maximum number of emails that may be processed per minute. The default is 30.

- **Send Email Timeout (milliseconds)**: The timeout in milliseconds for sending emails via the Alfabet Server. The Alfabet Server wait the defined time for the SMTP server to send out outgoing emails. If the timeout elapses before the emails can be sent, the sending of emails will fail.

- Define the mode for the specification of the sender email address of emails sent by the system by means of the **System Sender Email Account** and **Failover Sender Email Account** parameters as described in the section [Specifying Sender Email Addresses](#).

> ⚠️ The **System Sender Email Account** and **Failover Sender Email Account** parameters must also be defined in the server alias of the Alfabet Web Application.

4) Click **OK** to save your changes.

Testing the Connection to the SMTP Server

The Alfabet Administrator allows you to test the connection to the SMTP Server independent of a specific Alfabet functionality. The functionality must be tested on the server alias of the Alfabet Server connecting to the SMTP server.

> ⓘ The functionality to check the connection to the SMTP server is only available if event queueing is used for execution of processes via the Alfabet Server. The menu option will only be active if the **Remote Alias for Connection to the Application Server** attribute in the **Application Server** tab of the server alias is empty.

1) In the explorer of the Alfabet Administrator, right-click the server alias for which you want to test the email dispatch and select **Connect** in the context menu. A login window is displayed.

2) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click *OK* to connect to the Alfabet database.

3) Right-click the server alias again and select **Check SMTP**.

4) In the dialog box that opens, the SMTP server and the sender email address used to send the email are displayed. The sender email address is specified in the AlfabetMS.xml as the system mail account. Define the following in the **SMTP Test Dialog** editor:

- **Send to Email**: Enter the recipient's email address.

- **CC**: Optionally enter the email address of a recipient that should receive a copy of the email.

- **BCC**: Optionally enter the email address of a recipient that should receive a copy without being visible to the other recipients of the email.

- **Subject**: Enter the text that will appear in the subject line of the email.

- **Body**: Optionally enter the email text.

5) Click the **Send** button. The email is dispatched.

6) Review the availability of the email in the recipient's email account.

Specifying Sender Email Addresses

Typically, an SMTP server as well as the Microsoft Graph API of Azure deployments requires a sender email address for emails to be sent out. The sender address used for outgoing emails sent in the context of Alfabet by the Alfabet components will depend on the setting of the **System Sender Email Account** attribute in the **Server Settings** > **Email Settings** tab of the server alias configuration.

The sender email address is written into the ALFA_EMAIL_BUS database table when scheduling the email for sending. If a user triggers sending of an email via the Alfabet user interface, the information is added to the table by the Alfabet Web Application. If for example a workflow is started via an event, the Alfabet Server triggers sending of emails.

Therefore, the following attributes must be set in the server alias of both the Alfabet Web Application and the Alfabet Server:

- If the **System Sender Email Account** attribute is defined, the specified email address will be used as the sender address for all emails even if the originator of the email can be evaluated.

- If the **System Sender Email Account** attribute is not defined, the email address of the user that is the originator of the email will be used as sender address. In this case, a failover email address should be defined via the **Failover Sender Email Account** attribute. Otherwise, sending emails will fail if the originator cannot be determined.

> For information about how to specify an email address for a user, see the section Creating a New User.

The sender address will be determined in the context of the functionality for which the email is sent. For example:

- **Assignment functionality**: If a notification is sent out to either the assignee or the originator of an assignment because a user has performed changes to the assignment via the Alfabet interface, the email address of the Alfabet user performing the changes is used as the sender email address. If a notification email is sent out via a batch job that reminds users that assignments are nearing their target date, the batch job will use an artificial email address.

- **Workflow functionality**: The sender is generally determined as follows:

  1) Email address configured in the **Notification Sender Address** attribute for the workflow template.

  2) Email address of the current user performing the action.

3) Email address of the workflow owner.

4) Email address of the workflow template owner.

If the email address of the originator is not specified in the user configuration data of the originator of the email, the setting of the **Failover Sender Email Account** attribute of the server alias is evaluated:

- If the **Failover Sender Email Account** attribute is set, the specified failover email address will be used as sender address.

- If the **Failover Sender Email Account** attribute is not set, the sending of emails will fail.

Please note the following:

- In Microsoft Azure, the sender email must be an account that exists in the configured Microsoft Azure tenant.

- The **Failover Sender Email** and **System Sender Email Account** specification can be defined in the Alfabet user interface in the **Override Alfabet Settings** (CONF_Override_Settings) functionality and activated during testing. An activated definition in the **Override Alfabet Settings** functionality supersedes the settings in the server alias.

Re-Routing Emails to a Defined Address for Testing

When testing Alfabet functionality including sending of emails, you can configure the Alfabet Web Application to send all emails to a specified test email account, thus preventing the email from being sent to Alfabet users. This allows testing to be performed without changing the email account configuration of the Alfabet users.

To ignore email account settings of Alfabet users and to send all emails to one central email account, you can configure the server alias of the Alfabet Web Application and the Alfabet Server.

Alternatively, a test email account specification for the Alfabet Web Application can be defined in the user interface in the **Override Alfabet Settings** (CONF_Override_Settings) functionality and activated during testing. An activated definition in the **Override Alfabet Settings** functionality supersedes a setting in the server alias.

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer.

2) In the table, select the server alias that you want to edit and click the **Edit** button in the toolbar. An editor opens.

3) In the **Server Settings** tab, enter the name of the test email account in the **Test Receiver Email Account** field.

4) Click **OK** to save your changes.

After performing the test, delete the email address from the **Test Receiver Email Account** field.

If the **Test Receiver Email Account** field is empty, the emails are sent to the email account specified in the Alfabet user configuration for the Alfabet user that is the recipient of the email.

Configuring the Setup To Open the Alfabet Interface via Links in Email Notifications

Most emails generated in the context of Alfabet capabilities include hyperlinks to relevant views in the Alfabet interface. The following setup is required to ensure that access to Alfabet views is properly implemented:

- Access permissions must be granted to users according to the rules described in this section.

Access Permissions To Open the Alfabet User Interface

Access permissions of the users accessing the Alfabet interface via an email link depend on the following:

- The message type of email.

- The configuration of the hyperlink in the text template

- The **Allow Anonymous User** and **External Access** attributes in the server alias configuration of the Alfabet Web Application.

- The authentication method used in the company.

> ⓘ For an overview of the access modes for single sign-on or standard login with user name and password as well as an explanation about the concepts of access for named and anonymous users, see the section Configuring User Authentication.

Two message types are implemented in Alfabet for emails: work messages and Web messages. The table below provides information about the differences between the two types of messages and lists the features for which the messages are sent out:

Message Type	Characteristics	Sent In the Context of:
Web Message	The express view is sent for informational purposes only. The recipient of the email can be any person and does not have to be a Alfabet user.	- Express View
Work Message	Work messages are sent out to defined Alfabet users and typically inform users about changes to an object or tasks that must be fulfilled for an object.	- Assignments - Workflows - Monitors - Discussions - Organizational or Business Process Changes

The following information is available:

- [Access Permissions for Alfabet Access from Work Messages](#)

- [Access Permissions for Alfabet Access from Web Messages](#)

Access Permissions for Alfabet Access from Work Messages

If a hyperlink to a relevant Alfabet view is sent in a work message, the recipient can access the view with Read/Write access permissions to the view as long as the user profile used to open the view has Read/Write access permissions defined. The **Home** button is not displayed when a user opens the user interface via a link from a work message (even for named users). Depending on how the **Use Recipient's User Profile for External Links** attribute is set, either the user profile of the sender or of the recipient of the view will be used to open the user interface.

> ⓘ For more information about the configuration of user profile access permissions, see the chapter *Configuring Access Permissions for Alfabet* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- If single sign-on is used to authenticate the user, authentication is done automatically and the Alfabet view will be directly displayed to the user. If the user is not configured in the Alfabet database or is of the user type `Anonymous`, the view opens with ReadOnly access permissions.

- If the standard login is used, a login screen will be displayed once the user clicks the link. Authentication with user name and password will be required.

Access Permissions for Alfabet Access from Web Messages

If the hyperlink to the Alfabet view is sent in a Web message, the **Allow Anonymous User** checkbox in the server alias configuration of the Alfabet Web Application must be defined to allow access to the user interface via the link and the **External Access** attribute must be defined either as **Allowed as Visitor** or **Allowed as Authenticated User**.

If **Allowed as Visitor** is selected, the user interface will open with ReadOnly access with the temporary user `Visitor` as the login user. This access mode is always used even if the person using the link is a named Alfabet user. The **Use Recipient's User Profile for External Links** attribute is ignored, and the view always opens with the user profile of the sender of the express view mail, but ReadOnly.

If **Allowed as Authenticated User** is selected, login to the user interface is required. Anonymous users can only log in if single sign-on is used for login. With standard login, a login screen is displayed and a user name and password for a named user is required. Depending on how the **Use Recipient's User Profile for External Links** attribute is set, either the user profile of the sender or of the recipient of the view will be used to open the user interface.

To define whether login is activated for Web messages, you have to configure the server alias of the Alfabet Web Application:

1) Click the **Alfabet Aliases** node in the **Administrator** explorer. A table is displayed in the work area.

2) In the table, click the server alias that you want to configure access permissions for.

3) In the toolbar, click the **Edit** button. An editor opens.

4) In the **Server Settings < Security** tab, select the **Allow Anonymous User** checkbox to allow access to Alfabet views from Web messages. If you deselect the checkbox, access will be denied for all users including named users.

5) Select the way the interface will open if someone accesses the Alfabet user interface via a link in a web message in the **External Access** field. The following options are available:

- **Not Allowed**: Access to the Alfabet user interface via links in Web messages is disabled.

- **Allowed as Visitor**: the user interface opens with ReadOnly access with the temporary user `Visitor` as the login user, even if the recipient of the express view email is a named user.

- **Allowed as Authenticated User**: Login to the user interface is required. Anonymous users can only log in if single sign-on is used for login. With standard login, a login screen is displayed and a user name and password for a named user is required.

Defining the User Profile Used To Open the View

User profiles have a significant impact on the available functionality of a Alfabet view. For example, attributes of object classes can be visible and editable in one user profile and hidden or non-editable in another user profile. The user profile used by the recipient to open a Alfabet view is, by default, a user profile of the recipient:

- If the recipient of the email has more than one user profile assigned, the user profile specified as the default user profile in the user settings will be used to open the view. For more information about the definition of user settings, see the section *Defining Your User Settings in Alfabet* in the reference manual *Getting Started with Alfabet*.

- If a default user profile has not been specified in the user's user settings, the last user profile specifying Read/Write access permissions in the **Default User Profile** attribute in the **User Settings** editor will be used to open the view.

- If the user has no user profiles with Read/Write permissions assigned, the last user profile specifying ReadOnly access permissions in the **Default User Profile** attribute in the **User Settings** editor will be used to open the view.

- If the user is an anonymous user or has no user profiles assigned, the user profile defined as the default user profile for anonymous access will be used to open the view. If no user profile is defined for anonymous login, anonymous users cannot open the view.

ⓘ Please note that opening the view with the recipient's user profile might change the content of the view according to the permissions and settings that apply to the recipient's user profile, but it will not prevent access to the complete view that has been sent via the express view functionality. For example, a configured report that is configured to be visible for a specific user profile only can be opened as an express view with other user profiles. It is the intention of the express view functionality to provide a view to a colleague for a specific reason even if this view is normally not in the range of the colleagues' responsibilities.

However, you can configure the Alfabet Web Application to use the sender's user profile.

⚠ It is recommended the Alfabet Web Application is configured to open Alfabet views using the recipient's user profile in order to prevent violation of access permission configurations. For example, if you are logged in with an administrative user profile while sending an email link to another user, and the Alfabet view opens with the sender's user profile, the recipient will have Read/Write

access permissions to all objects regardless of the configured access permissions in Alfabet and will see all objects regardless of the mandate settings.

For more information about administrative user profiles, see *Creating a User Profile* in the reference manual *User and Solution Administration*.

To configure the Alfabet Web Application to open views using the sender's user profile, deselect the **Use Recipient's User Profile for External Links** checkbox in the **Server Settings** tab of the **Server Alias** editor of the Alfabet Administrator.

ⓘ Please note the following:

- If no Alfabet user can be evaluated as the sender of the email, the Alfabet interface will open with the recipient's user profile even if the **Use Recipient's User Profile for External Links** checkbox is not selected. Alfabet batch utilities, the view opens with the user profile of the specified user. For users with more than one user profile attached, the user profile is selected as described above for the recipient's user profile.

- If the interface is opened via a link in an express view and the Alfabet Web Application is configured to allow users as visitors, the setting of **Use Recipient's User Profile for External Links** checkbox is ignored, and the user profile of the sender is used with ReadOnly access permissions.

Permissions to Edit Objects

When a user accesses a view via a hyperlink in a Web message, he/she can only edit the object if the **External Access** attribute is set to **Allowed as Authenticated User**, the user profile used to open the view will grant ReadWrite access permissions and the current user will have Write permissions for the object that the view is about. If the **External Access** attribute is set to **Allowed as Visitor**, the user interface will always open ReadOnly.

When a user accesses a view via a hyperlink in a Work message, access permissions depend on the access permissions granted by the user profile used to display the view and the access permissions granted to the current user for the object the view is about.

ⓘ For an overview of the access permissions that define the edit rights to objects in the Alfabet database, see the section *Configuring Access Permissions for Alfabet* in the reference manual *Configuring Alfabet with Alfabet Expand*.

For example, a named user opening a Alfabet view via a hyperlink can have edit rights to an object because he/she is member of an authorized user group or because an access permission rule is defined in the configuration tool Alfabet Expand that grants him/her edit rights to the object. If the Alfabet view opens via a user profile with ReadWrite access permissions, the user will then be able to edit the object.

Navigation Options for Users Accessing Alfabet via a Hyperlink

If a user accesses a Alfabet view via a hyperlink in an email, neither the navigation bar of the user profile used to display the view, nor the button "Home" will be available. Nevertheless, the user can navigate to other views in the Alfabet interface from the view sent via an email. This is possible, for example, by

opening a page view in an object view and double-clicking an object displayed in the page view. Whether the user can edit the object will depend on the access permissions configured for the user profile used to display the view and the access permissions granted to the current user for the object that the view is about.

Configuring the Express View (Email) Capability

The express view capability in Alfabet allows users to send a hyperlink for an Alfabet view via email to an-other person. The person receiving the email does not have to be a Alfabet user.

An express view can be sent for any view for which the **Mail Express View** functionality is available in the **Bookmark** menu of the main toolbar of the Alfabet user interface.

> (i) The button can be disabled for specified configured reports and standard Alfabet views via the view configuration.

When the recipient of the email clicks the hyperlink in the email, the Alfabet user interface will open in the standard browser of the recipient and the user will have ReadOnly access to the view.

> (i) If the user opening the view is already working with Alfabet, the same session will be used to open the view received via email. The user can change to the user profile he/she was originally logged in with using the **Change User Profile** option in the <AlfabetUser> menu on the top left of the user interface.

The following setup is required to implement this functionality:

- The dispatch of emails must be activated as described above. For more information, see the section Configuring the Alfabet Server to Connect to the SMTP Server.

- The access permissions to access the Alfabet interface via hyperlinks sent in Web messages must be configured. For more information, see the section Configuring the Setup To Open the Alfabet Interface via Links in Email Notifications.

In the server alias of the Alfabet Web Application, define how express views shall be sent:

1) Click the **Alfabet Aliases** node in the **Administrator** explorer. A table is displayed in the work area.

2) In the table, click the server alias of the Alfabet Web Application.

3) In the toolbar, click the **Edit** button. An editor opens.

4) In the **Server Setting** > **Security** tab, select the **Allow Anonymous User** checkbox (=True). If anonymous access is not activated, only named users with access permissions granted by single sign-on can open the link.

5) In the **Server Setting** > **General** tab, select the **Send Web Message Direct** checkbox if Web messages should be sent directly from Alfabet to the user who is creating the message associated with a page view. If you do not select this checkbox, a window opens when a user uses the express view functionality. The correct express view link to the Alfabet user interface is displayed and can be copied to any email. Alternatively a link is provided that opens an email containing the express view link via the default email client on the client host.

Activating the History Tracking Functionality

Alfabet provides a history tracking functionality that documents the changes made to an object. All changes made to an object's standard and custom properties and relations are documented.

The history tracking functionality is implemented on a class by class basis. Audit information is stored in a separate table for each object class that history tracking is enabled for. For each object class a table named <name of database table for object class>_AU is created and information about the existing objects is added to the table to initialize the audit. Whenever an object is changed, a new line is written to the audit table.

The audit table stores information about the object that has been changed, the kind of change performed to the object, the time the object was changed and the user that performed the change.

By default the user name of a user performing the change is stored in the audit table. This information is then visible to all users that view the history of the object in the Alfabet user interface. If you do not want the user names of users to be stored in the audit table, you can optionally configure the Alfabet Web Application to store the technical name of the user instead of the user name in the audit tables. The technical name is an attribute in the user configuration in Alfabet that is exclusively used for the audit history. Please note that for changes performed via processes like ADIF import, ALFABET_SYSADMIN will be used as user information in the audit table if no user information is returned in the context of the process. This is for example the case for auto-run ADIF jobs executed during update of the meta model via the Alfabet Administrator, because execution of processes via the Alfabet Administrator is done with access to the database via a database user without specification of an Alfabet user.

⚠ Auditing is performed by database triggers. When the Alfabet Web Application is restarted, Oracle database servers may set the audit triggers to the state invalid and show a warning when opening the database manager. This change in the trigger state does not have any effect on the trigger functionality. Audit triggers are automatically reset to valid the first time the trigger is used after restart of the Alfabet Web Application. The warning in the Oracle database manager can be ignored.

ⓘ For information about viewing the audit history of an object, see the section *Viewing the Change History of an Object* in the reference manual *Getting Started with Alfabet*.

For information about the structure of the database table for auditing, see the section *Defining Audit Management Related Configured Reports* in the reference manual *Configuring Alfabet with Alfabet Expand*.

The following is required to activate the history tracking functionality.

Application Administrator

Activate history tracking for Alfabet components. For the Alfabet Server, history tracking is activated in the server alias configuration as described below in the section Activating History Tracking for the Alfabet Server .

For the Alfabet Web Application, history tracking is activated via the context menu of the server alias node as described below in the section Activating History Tracking for the Alfabet Web Application .

By default, history tracking is deactivated.

Optionally, you can configure the Alfabet components to use the technical name of the user instead of the user name for auditing.

For more information, see [Configuring User Information Displayed in History Tracking](#).

Solution
Designer

Activate audit per object class by setting the **Audit** attribute of the object class to `True`.

By default, all relevant IT classes have the history tracking functionality activated.

For more information see the section *Specifying History Tracking for an Object Class* in the reference manual *Configuring Alfabet with Alfabet Expand*.

Activating History Tracking for the Alfabet Server

To activate the history tracking functionality for the Alfabet Server:

1) In the Alfabet Administrator, click the **Alfabet Aliases** node in the **Administrator** explorer. A table is displayed in the work area.

2) In the context menu, click the server alias of the Alfabet Server that you want to configure history tracking for.

3) In the toolbar, click the **Edit** button. An editor opens.

4) In the **Server Settings** tab, select the **Enable Audit** checkbox in the **General** tab to activate history tracking for all object classes with the **Audit** attribute set to `true`. If you deselect the checkbox, auditing will be deactivated for all object classes independent of the setting of the **Audit** attribute.

5) Restart the Alfabet Server. The definition of the **Enable Audit** attribute requires the Alfabet Server to be restarted in order for the specification to be technically implemented.

Activating History Tracking for the Alfabet Web Application

To activate the history tracking functionality for the Alfabet Web Application:

1) In the Alfabet Administrator, click the **Alfabet Aliases** node in the **Administrator** explorer. The node is expanded.

2) Right-click the server alias of the Alfabet Web Application that you want to configure history tracking for and select **Connect**.

3) Log in to the database server. The connection is established.

4) Right-click the server alias of the Alfabet Web Application that you want to configure history tracking for and select **Activate Audit**.

In the context menu, there is also a **Deactivate Audit** option to deactivate an active audit. As long as the audit is deactivated, no entries are written to the audit tables.

Configuring User Information Displayed in History Tracking

To configure the history tracking functionality to use the technical name of a user instead of the user name:

1) Click the **Alfabet Aliases** node in the **Administrator** explorer. A table is displayed in the work area.

2) In the table, click the server alias of the Alfabet Web Application that you want to configure history tracking for.

3) In the toolbar, click the **Edit** button. An editor opens.

4) In the **Server Settings** tab, select TECH_NAME in the **Update History User Name** field in the **General** tab.

5) Expand the **Alfabet Aliases** node in the **Administrator** explorer.

6) In the explorer, right-click the server alias that you want to configure history tracking for an select **Connect**. A login window is displayed.

7) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database. The explorer node is expanded.

8) Expand the User Management node and click Users. A table listing all users is displayed in the work area.

9) Check whether the technical name of the user, displayed in the column **Tech Name**, is set for all users. If the technical name of a user is not set, click the **Edit** button, enter a technical name for the user in the **Technical Name** field of the **User** editor and click **OK** to save your changes.

ⓘ The Tech Name property of the user is entered instead of the user name in the columns CREA-TION_USER and LAST_UPDATE_USER of the audit history tables. These columns are displayed in the standard history. The columns DELETE_USER and AUDIT_USER are still filled with the User Name instead of the Tech Name property.

About Batch Utilities for Alfabet

Software AG provides several batch utilities that must be executed to activate some Alfabet functionalities, to enhance the usability of some Alfabet functionalities, or to update data in the Alfabet database. Please note however that it is recommended to schedule batch processing via the **Job Schedule** functionality on the Alfabet user interface which allows regular execution of batch jobs without running a separate executable.

The following applies to all batch utilities:

- By default, the executables for batch job processing are all located in the `Programs` sub-directory of the Alfabet installation directory.

- All executables have a mandatory configuration file named `<executable name>.exe.config` which must be located in the same directory as the executable to execute batch jobs.

- Executables for batch jobs can be started in a command line or by means of a Windows® batch job.

  When executing a Windows batch job, the execution time for a batch job is defined with the help of the Windows scheduler for batch jobs. When starting the executable with a command line, the batch job is executed immediately.

- The command line options for the tools are listed in the following chapters with the description of the individual tools. If a command line option that requires a parameter definition and the parameter includes whitespace, the parameter must be written in inverted commas, e.g.

```
Executable.exe-statement "this is the statement"
```

Server Alias Configurations Determining the Processing Mode for Batch Jobs

The way batch jobs are executed depends on the setting of the attributes in the Application Server tab of the server alias configurations of the Alfabet components:

- If **Use Event Queue for All Jobs** is selected, all server processes are scheduled via event queueing. This is the recommended option that will be maintained in future releases.

- If **Use Application Server and Net Remoting Service** is selected, processes are directly handed over to the Alfabet Server for execution via remote service calls. The command line tools can then be either started via a remote alias configuration to connect to a running Alfabet Server or can act directly on the Alfabet database via a server alias configuration. This method will be deprecated in the future because it will not be compatible with the upcoming.NET Core component.

The setting in the **Application Server** tab must be identical in all server alias configurations of all Alfabet components including batch utilities.

Standard Logging for Alfabet Batch Utilities

A standard logging method exists for Alfabet utilities. Logging information is written to a log file at runtime in the following format:

```
<date and time> <message type> <message text>
```

For example:

```
2009-02-24T12:30:15.4397058Z INFO start external source
synchronization
```

The timestamp is the UTC time (coordinated Universal Time) and may therefore differ from the time in your local time zone. The timestamp is written in ISO 8601 combined date and time format as year-month-day-Thour:minutes:secondsZ.

The message type can be one of the following:

- **ERROR**: An error occurred. The message describes the type of error.

- **WARNING**: Problems were encountered during the execution of the utility that are not as severe as an error. The process was executed but the result should be checked. The message describes the problem.

- **INFO**: Information about the normal execution of the utility is given.

Command line options that can be used with every Alfabet utility using the standard logging allow you to specify the number and location of log files as well as the amount of detail of the logging information:

Command Line Option	Default	Description
-logpath <log file path>	Log file is stored in the working directory of the executable.	Specification of a path to a directory for storage of the log file.
-logfile <log file name>	<Executable><timestamp>.log	Specification of the log file name. Allowed file extensions are LOG and TXT. **NOTE**: Setting -logfile changes the way log information is stored. If -logfile is not defined, a new log file is created each time the utility is used and the log file name is extended with a timestamp specifying the current UTC time. The timestamp is the UTC time written as "_yyyymmdd_hhmmss". For example: logfilename_20090328_150827.log. If -logfile is defined, logging information is appended to the already existing log file each time the utility is used. To restrict the file size, you can set the -logclear option to delete old log messages.
-nologappend		This option is only evaluated if -logfile is defined.

Command Line Option	Default	Description
		If `-nologappend` is set, the log file defined with `-logfile` and `-logpath` is overwritten with a new file containing the current log information each time the utility is used. If `-nologappend` is not set, logging information is appended to the already existing log file each time the utility is used. **NOTE**: To restrict the file size, you can set the `-logclear` option to delete old log messages.
`-logverbose`		If `-logverbose` is set, additional information about the running process is logged. The content and amount of additional information messages depend on the utility used. If setting `-logverbose` does not change the log output, there are no additional information messages available for the utility. **NOTE**: Verbose logging is in most cases not required and can lead to a decrease in performance.
`-logclear <number of days>`	Infinite	This option can only be used if `-logfile` is defined and `-nologappend` is not set. During logging, the log file is scanned for log messages with a timestamp older than the number of days specified with `-logclear` and these messages are deleted. **NOTE**: The scanning process can lead to drawbacks in performance.

Running Batch Jobs with Encrypted Command Line

Command line parameters can be encrypted, and the call can be made with the encrypted parameters for security reasons, for example to secure command line information in a Windows batch job. To encrypt the parameters in a command line, the command can be encrypted with the following command:

```
<NameofExecutable>.exe-encodeparams <full set of command line parameters to
run executable>
```

For example:

```
AlfaBatchExecutor.exe-encodeparams-msalias Production-alfaLoginName
Administrator-SystemDateMonitor
```

This command returns the complete encoded call that must be used to run the executable with encoded parameters without the "Please use: " at the beginning of the returned string. In general, the call structure to execute the process with encrypted command line is:

```
<NameofExecutable>.exe-encodedparams <encrypted parameters>
```

Please note that all parameters in the command line must be encrypted. It is not possible to run commands with a mixture of encoded and non-encoded parameters.

For the example above, the command would return:

```
Please use: AlfaBatchExecutor.exe-encodedparams
YAyyXYPeVgEm4VDCNZ9Mfy9Whu2Lyo43v8zemWffqFUJB3MXCRp9MOCrjmklWMXT/h64G
cDlkj9kMbyy7ff05NLz6W5eLADYguRan9/SEGE=
```

To run the command with executed parameters, run:

```
AlfaBatchExecutor.exe-encodedparams
YAyyXYPeVgEm4VDCNZ9Mfy9Whu2Lyo43v8zemWffqFUJB3MXCRp9MOCrjmklWMXT/h64G
cDlkj9kMbyy7ff05NLz6W5eLADYguRan9/SEGE=
```

An encoded command line can be decoded with the following command:

```
<NameofExecutable>.exe-decodeparams <encrypted parameters>
```

Batch Tools Available for Activating and Executing Alfabet Functionality

This section describes in detail the functionality and handling of the batch tools delivered with the Alfabet components.

- [Triggering Target Date Control for the Assignments Capability](#)

- [Batch Processing for Monitors and Change Management with AlfaBatchExecutor.exe](#)

- [Updating Indexes with the FullTextSearchUtil.exe](#)

- [Batch-Calculation of Indicators with RescanIndicatorsConsole.exe](#)

  - [Defining the Data To Be Calculated in a Configuration File](#)

  - [Running RescanIndicatorsConsole.exe](#)

- [Batch Evaluation of Color Rules with RescanColorRules.exe](#)

- [Batch Processes for Workflows with AlfaWorkflowCommandPrompt.exe](#)

- [Batch Processes Relevant for the Alfabet Publication Framework](#)

  - [Triggering Publication via a Batch Utility](#)

    - [Providing Relevant Data and Configuration](#)

    - [Starting the Publication Console Application](#)

    - [Defining Publication Output in the Command Line of the Batch Utility](#)

  - [Deleting Expired Publications from the Database](#)

Triggering Target Date Control for the Assignments Capability

The assignments capability in Alfabet allows users to collaborate with one another about objects in the enterprise architecture. An assignment is a task that is defined for a selected object and assigned to a specific user. The assignee must provide the required input for the object by a specified due date.

The assignment functionality is preconfigured and basic functionalities can be implemented with default settings. The basic functionalities can optionally be configured by a solution designer in the tool Alfabet Expand.

- For general information about the use of assignments in Alfabet, see the section *Sending and Receiving Assignments for Alfabet Objects* in the reference manual *Getting Started with Alfabet*.

- An overview of the configuration for the assignment capability is provided in the section *Configuring the Assignment Capability* in the reference manual *Configuring Alfabet with Alfabet Expand*.

While many of the functionalities available for the assignment capability are triggered by user interaction when a user creates or changes an assignment via the Alfabet user interface, the control of target dates defined for assignments is triggered via a batch job. The batch job must be run in regular intervals.

The batch job for triggering the control of target dates does the following:

- Checks assignment target dates and, if the reminder period defined for the target date has begun, sends notification emails to the assignee. Notifications will only be sent to users if they have explicitly selected to be informed via email when the reminder period is reached.

- Checks assignment target dates and 1) closes and removes any optional assignments from the user's views that have reached the target date and 2) returns any mandatory assignments to the originator of the assignment. email notifications will be sent to the originator when an assignment is returned.

Note the following for the execution of the batch job:

- Target dates are specified in days. It is sufficient to run the batch job once a day to process all reminders and return/close assignments in a timely manner.

- Emails are only triggered via batch job if the assignment capability is configured to send notification emails. Assignments that have surpassed the target date will be returned or closed via the batch job regardless of whether the email capability is implemented.

> The following is required to activate the sending of notification emails:
>
> - Activate sending emails for assignments in Alfabet Expand. For more information, section *Configuring the Assignment Capability* in the reference manual *Configuring Alfabet with Alfabet Expand*.
>
> - Configure the Alfabet Web Application to connect to an SMTP server and set the required parameters necessary to open Alfabet views via the hyperlinks in email notifications. For more information, see the sections Configuring the Alfabet Server to Connect to the SMTP Server to the SMTP Server and Configuring the Setup To Open the Alfabet Interface via Links in Email Notifications.

- Both checks listed above are executed simultaneously. As a consequence, if a user specifies a reminder period after the target data (in other words, enters a negative number for the reminder period), reminder notifications will not be sent.

- If you want the email recipient to open the Alfabet view using the sender's user profile, you must specify the optional parameter `-alfaLoginName` in the command line starting the tool. The view will then open with the user profile specified in the personal settings of the user specified with `-alfaLoginName`. If the parameter is not set when executing the batch job, the view will open with the recipient's user profile even if the Alfabet Web Application is configured to use the sender's profile to open views from email links.

- If an email address is specified in the **System Sender Email Account** parameter of the server alias configuration, the batch utility uses that email address as sender address in the notification emails. Otherwise the batch utility uses the email address `AssignmentManager@Alfabet.system`. The recipient cannotreply to this email address.

- Depending on the configuration of the assignment capability in Alfabet Expand reminders send within one batch job to the same recipient are either send as one email per reminder, or are concatenated within a single email.

Batch jobs are executed with the executable `AlfaBatchExecutor.exe`.

Executable	`AlfaBatchExecutor.exe` located in the ***Programs*** subdirectory of the Alfabet in-stallation directory
Remote or stand-alone ac-cess	Stand-alone access to the database with a server alias or remote access with a re-mote alias connecting to a running Alfabet Server.
Preconditions	An `AlfabetMS.xml` configuration file with a server alias configuration specifying the connection to the Alfabet database is available and the respective functionality is configured.
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see <u>Standard Logging for Alfabet Batch Utilities</u>.
Command line help	Start executable with `-h` or `-help`

The executable must be started with the following parameters:

```
alfaBatchExecutor.exe-msalias <alias name>-alfaLoginName <user name>-
alfaLoginPassword <user password>-jobClass AssignmentJob
```

The table below displays the command line options:

Command Line Option	Mandatory/ Optional	Explanation
`-jobClass`	Mandatory	Enter `AssignmentJob` to execute target date control for assignments.
`-msalias <alias name>`	Mandatory	Enter the alias name as specified in the `AlfabetMS.xml` configuration file. If a remote alias is specified, the corresponding Alfabet Server must be running. If a server alias is specified, all other Alfabet applications with access to the Alfabet database must be stopped.
`-msaliasesfile <Alfabet configuration file path>`	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName`	Mandatory	User name for access to the Alfabet database. When the Alfabet Server is configured to use the sender's user profile to display the Alfabet user interface when accessed by a hyperlink in the email, the `alfaLoginName` is also used to identify a sender profile to open hyperlinks to Alfabet views from the notification emails sent out by the batch job. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPassword`	Optional	Password for access to the Alfabet database.
`-optionalArgs`	Optional	Only when specified for the batch job type. You can enter the optional arguments in a comma-separated list here.

Batch Processing for Monitors and Change Management with AlfaBatchExecutor.exe

Software AG provides batch processes to support the execution of defined Alfabet processes. The batch processing tool works with a remote alias so that it is possible to run the batch job while the Alfabet Server is running.

Via a batch job, assignments are created for users and/or automatic email notifications will be sent to an object's authorized user if a monitor is triggered, an assignment approaches or reaches a defined due date, or an organizational or process change has occurred that affects an object.

The execution of the batch job in regular intervals is mandatory in order to activate the function-alities described above. For example, if a user specifies a monitor for an object, he/she specifies the monitored context and the group of listeners that are notified by email about the monitor, and the frequency that the monitor should be executed (for example, a week before a targeted deadline). However, these specifications alone do not activate the monitor. A batch job must be executed in the specified frequency interval in order to actually execute the monitor. Without the batch job, the monitor will remain inactive. It is therefore recommended that you run the batch job at least once a day to ensure that, for example, all specified monitors are executed in the given frequency.

When executing a batch job, log messages are written to a log file. You may see messages about the successful execution of monitor jobs even if no action was performed because all configured monitors are not due. In the log file, execution of a monitor is specified as the whole process of checking the monitor.

The following types of batch jobs are available for batch execution:

- **ObjectDateMonitor** for the execution of date monitors specified by Alfabet users. The batch job generates email notifications triggered by the monitor.

- **ObjectActivityMonitor** for the execution of activity monitors specified by Alfabet users. The batch job generates email notifications triggered by the monitor.

- **ObjectInactivityMonitor** for the execution of inactivity monitors specified by Alfabet users. The batch job generates email notifications triggered by the monitor.

    - For general information about the implementation of activity, inactivity, and date monitors, see the section *Keeping Track of Objects via Monitors* in the reference manual *Getting Started with Alfabet*.

    - The object classes and their monitor contexts for which the monitor functionality is available must be defined in the configuration tool Alfabet Expand. For more information, see section *Configuring Monitors* in the reference manual *Configuring Alfabet with Alfabet Expand*.

    - The text in email notifications is also determined in the configuration tool Alfabet Expand. For more information, see section *Configuring Text Templates for Email Notifications* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- **ConsistencyMonitor** for the execution of consistency monitors. The batch job executes the Alfabet queries defined for the consistency monitors. Assignments are automatically generated for each object found by the Alfabet query. Email notifications can additionally be generated via batch process for the consistency monitor..

    - The text templates used for the assignment description are based on the text template configured in the **M_CON** text template folder in the configuration tool Alfabet Expand. The relevant text template must be selected in the **Consistency Monitor** editor in the **Consistency Monitors** functionality in the **Admin** application. For more information, see the section *Configuring Monitors* in the reference manual *User and Solution Administration*.

    - The text template used for the email notification is the text template **ConsistencyMonitorMail** located in the **MON** text template folder in the

configuration tool Alfabet Expand. The relevant text template must be selected in the **Consistency Monitor** editor in the **Consistency Monitors** functionality in the **Admin** application. For more information, see *Configuring Monitors* in the reference manual *Configuring Alfabet with Alfabet Expand*

- *NotificationMonitor* for the execution of notification monitors. The batch job executes the Alfabet queries defined for the notification monitors. Email notifications are automatically generated for each object found by the Alfabet query.

    - For general information about the configuration and implementation of notification monitors, see the section *Defining Notification Monitors* in the reference manual *User and Solution Administration*.

- *SystemDateMonitor* for the execution of system-wide date monitors. The batch job generates email notifications triggered by the monitor. This is specified by administrator in the System Date Monitors functionality. See

    - For general information about the configuration and implementation of system-wide date monitors, see the section *Defining System Date Monitors* in the reference manual *User and Solution Administration*.

    - The assignments generated in the context of system-wide date monitors require further configuration in the configuration tool Alfabet Expand. For more information, see the section *Configuring Assignments for System-Wide Date Monitors* in the reference manual *Configuring Alfabet with Alfabet Expand*.

    - The text in email notifications is also determined in the configuration tool Alfabet Expand. For more information, see *Configuring Text Templates for Email Notifications* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- *OrganizationalChange* for the dispatch of email notifications about the assignments generated due to organizational changes. An XML object for the definition of organizational changes must be configured in Alfabet Expand to execute this type of batch job. See *Configuring the Propagation of Organizational Changes* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- *BusinessProcessChange* for the dispatch of email notifications about the assignments generated due to business process changes. An XML object for the definition of organizational changes must be configured in Alfabet Expand to execute this type of batch job. See *Configuring the Propagation of Business Process Changes* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- *ExpiredReportDeletionJob* for the deletion of all expired APF publications from the **Internal Document Selector**. Publications are generated on basis of a publication definition based on a Microsoft® Word template. Publication can either be triggered by a batch job or by a Alfabet user opening a configured report that is defined to trigger publications. For more information about the configuration of APF publications, see *Publishing Data In Microsoft® Word Format* in the reference manual *Configuring Alfabet with Alfabet Expand*.

    An expiration date of one month after creation date is configured for publications. Once the expiration date is reached, the publication will no longer be displayed to authorized users in the Alfabet interface. However, it will remain in the **Internal Document Selector** until deleted with the utility `AlfaBatchExecutor.exe`.

Batch jobs are executed with the executable `AlfaBatchExecutor.exe`.

Executable	`AlfaBatchExecutor.exe` located in the ***Programs*** subdirectory of the Alfabet installation directory
Remote or stand-alone access	Stand-alone access to the database with a server alias. Remote access with a remote alias connecting to a running Alfabet Server is only possible for the batch processing of monitors.
Preconditions	An `AlfabetMS.xml` configuration file with a server alias configuration specifying the connection to the Alfabet database is available and the respective functionality is configured.
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see Standard Logging for Alfabet Batch Utilities.
Command line help	Start executable with `-h` or `-help`

⚠ The batch execution of monitors is only possible in the interval configured in the monitor with the parameter **Monitor Frequency**. Choices include Daily, Weekly, and Monthly. Weekly monitors are executed on the same day of the week that the monitor is created. For example, a monitor created on Thursday, March 27 will be executed on Thursday, April 3, Thursday, April 10, etc.

The executable must be started with the following parameters:

```
alfaBatchExecutor.exe-msalias<alias name>-alfaLoginName <user name>-
alfaLoginPassword <user password>-jobClass <type of batch job> [-
optionalArgs <optional Arguments for particular batch job>]
```

The table below displays the command line options:

Command Line Option	Mandatory/ Optional	Explanation
`-jobClass`	Mandatory	Enter the name of the batch job as given in the list above.
`-msalias` <alias name>	Mandatory	Enter the alias name as specified in the `AlfabetMS.xml` configuration file. If a remote alias is specified, the corresponding Alfabet Server must be running. If a server alias is specified, all other Alfabet applications with access to the Alfabet database must be stopped.

Command Line Option	Mandatory/ Optional	Explanation
-msaliasesfile <Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
-alfaLoginName	Mandatory	User name for access to the Alfabet database. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
-alfaLoginPassword	Optional	Password for access to the Alfabet database.
-optionalArgs	Optional	Only when specified for the batch job type. You can enter the optional arguments in a comma-separated list here.

Updating Indexes with the FullTextSearchUtil.exe

The executable `FullTextSearchUtil.exe` allows system administrators to create and update the search index for globally-defined full-text search groups. For more information about the configuration of search groups for Alfabet 's **Full-Text Search** functionality, see the section *Configuring the Full-Text Search Capability* in the reference manual *Configuring Alfabet with Alfabet Expand*.

ⓘ An index cannot be created or updated for object-centric search groups by means of the `FullTextSearchUtil.exe`. The user conducting the search creates and updates the index directly on the **Full-Text Search** page view for the object that the user wants to find related objects for.

The executable can be executed in client-server mode only. It is recommended that this executable is included in regular batch job queues.

ⓘ As an alternative to execution of `FullTextSearchUtil.exe`, Alfabet users with access to the **Job Schedule** functionality can schedule the creation and update of search indexes in defined intervals via a job schedule. For more information, see *Scheduling ADIF Jobs and Batch Jobs via the Job Schedule Functionality* in the reference manual *User and Solution Administration*.

Executable	`FullTextSearchUtil.exe` located in the **Programs** subdirectory of the Alfabet installation directory

Remote or stand-alone access	Remote access with a remote alias.
Preconditions	The Alfabet Server must be running.
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see Standard Logging for Alfabet Batch Utilities.
Command line help	Start executable with-h or-help

The executable must be started with the following parameters:

```
FullTextSearchUtil-msalias <remote MSAlias> [-msaliasesfile <file path to
Alfabet configuration file>]-alfaLoginName <user name> [-alfaLoginPassword
<user password>-searchGroupName <search group name>-language <locale ID>]
```

The table below displays the command line options:

Command Line Option	Mandatory/ Optional	Explanation
-msalias <alias name>	Mandatory	Enter the alias name as specified in the `AlfabetMS.xml` configuration file. If a remote alias is specified, the corresponding Alfabet Server must be running. If a server alias is specified, all other Alfabet applications with access to the Alfabet database must be stopped.
-msaliasesfile <Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
-alfaLoginName	Mandatory	User name for access to the Alfabet database. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
-alfaLoginPassword	Optional	Password for access to the Alfabet database.
-searchGroupName <search group name>	Optional	Specifies the search group for which the index is created. If-`searchGroupName` is not set, the utility will recreate the index for all search groups. The search group must be specified via the name as written in the XML attribute `Name` of the XML element `SearchGroup` in the XML object

Command Line Option	Mandatory/ Optional	Explanation
		SearchManager. If the name contains white spaces, the name must be written in quotation marks in the command line.
`-language` <locale ID>	Optional	Specifies the language for which the search index shall be created as locale ID such as 1033 for English(USA). The search index can be created for any language for which the **Support Data Translation** attribute is set to `True` in the Alfabet culture settings. For more information about culture settings, see *Specifying the Cultures Relevant to Your Enterprise* in the reference manual *Configuring Alfabet with Alfabet Expand*.

Batch-Calculation of Indicators with RescanIndicatorsConsole.exe

The tool `RescanIndicatorsConsole.exe` allows you to batch calculate indicators. Indicators can be calculated for the following:

- All indicators or all indicators of a defined evaluation type for a specific object class or object class stereotype.

- Indicators of a selected indicator type for a query-defined subset of objects from the Alfabet database.

Indicators are individually calculated per object in the **Evaluation** page view of an object class.

Do the following to rescan indicators:

- Optionally define which data should be calculated in an XML configuration file. The definition of the configuration file is only required when only a subset of indicators for objects found via a query will be calculated. To calculate all indicators or all indicators of a defined evaluation type for all objects of an object class, you can define the settings directly in the command line of the batch tool. For more information, see Defining the Data To Be Calculated in a Configuration File.

- Run the batch tool `RescanIndicatorsConsole.exe`. For more information, see Running RescanIndicatorsConsole.exe.

As an alternative to execution of `RescanIndicatorsConsole.exe`, Alfabet users with access to the **Job Schedule** functionality can schedule the batch calculation of indicators in defined intervals via a job schedule. For more information, see *Scheduling ADIF Jobs and Batch Jobs via the Job Schedule Functionality* in the reference manual *User and Solution Administration*.

During re-calculation of indicators, `RescanIndicatorsConsole.exe` substitutes existing indicators with new indicators by first deleting the existing indicator and then adding the indicator that results from the re-calculation. Therefore, if the re-calculation fails, the existing indicators

will be removed by the batch job. The log output of the `RescanIndicatorsConsole.exe` can be configured to log all exceptions for the re-calculation by setting the log method to `logverbose`. A log message is given per object/indicator combination. If a high number of indicators is re-calculated using the same failed algorithm, large log file sizes will result. It is recommended that indicator calculation methods are tested via the individual calculation of single indicators prior to running a batch job. The batch job should be started without verbose log mode to reduce log overhead.

> (i) Rescan of indicators may trigger a high number of single database insert and delete transactions. This can have a negative impact on fragmentation of indices. As a result, CPU usage is increased and performance is reduced. It is recommended to rebuilt indexes in regular intervals for object classes subject to rescan of indicators. This can be done using the command line tool `AlfaAdministratorConsole.exe`. For more information, see [Rebuilding Indexes on Database Tables](#) in the reference manual *System Administration*.

Defining the Data To Be Calculated in a Configuration File

The configuration file defining which indicators will be re-computed must be an XML file with the following structure:

```
<Config>
    <IndicatorEntry Name="Name" ClassName="ClassName"
    EvaluationType="EvaluationTypeName" IndicatorType="IndicatorTypeName">
        <Query><CDATAQuery></Query>
    </IndicatorEntry>
</Config>
```

XML Attribute	Mandatory/Optional	Explanation
`Name`	Mandatory	Define a name for the `IndicatorEntry`.
`Class-Name`	Mandatory	Specify the name of the object class that indicators will be re-computed for. You can either specify an object class or an object class stereotype. Object class stereotypes must be defined using the following syntax: *ClassName:StereotypeName* In the **Class Configuration** functionality, evaluation types can be assigned to object classes or object class stereotypes. If the computed indicators are assigned to the object class, specify the object class. If the computed indicators are assigned to the object class stereotype, define the object class stereotype. If you define the object class while computed indicators are assigned to the object class stereotypes only, no re-calculation will be done.

XML Attribute	Mandatory/Optional	Explanation
`EvaluationType`	Optional	Specify the name of an evaluation type to limit rescan of indicators to indicators assigned to the defined evaluation type.
`IndicatorType`	Optional	Specify the name of an indicator type to limit rescan of indicators to indicators assigned to the defined indicator type.
`Query (sub-element)`	Optional	Define a query in a child XML element `Query` of the XML element `IndicatorEntry` to restrict rescan of indicators to objects found by the query. The query must be defined in a CDATA element within the xML element `Query` and can be defined in Alfabet query language or native SQL. Please note that the correct syntax for definition of CDATA in XML must be used. **NOTE**: The query must find objects of the object class defined with the XML attribute `ClassName`. **NOTE**: When defining a native SQL query, the query must return the `REFSTR` of objects of the object class defined with the XML attribute `ClassName` of the XML element `IndicatorEntry` as only argument in the `SELECT` clause. When defining an Alfabet query, no show property definition is required.

Multiple XML elements `IndicatorEntry` can be defined. Setting most of the XML attributes is optional. The execution of an `IndicatorEntry` depends on the combination of attribute settings in the XML element `IndicatorEntry`:

Defined XML Attributes	Rescan of Indicators performed for...
`ClassName`	all indicator types that are defined for the object class or for the object class stereotype defined with the XML attribute `ClassName` are re-computed for all objects of the defined object class or object class stereotype.

`ClassName`
 	
`EvaluationType`
	all indicator types that belong to the evaluation type defined with the XML attribute ***EvaluationType*** will be re-computed for all objects of the object class or object class stereotype defined with the XML attribute `ClassName`.

`ClassName`
 	
`EvaluationType`
 	
`IndicatorType`
	indicators with the indicator type and evaluation type defined with the XML attributes ***IndicatorType*** and ***EvaluationType*** will be re-computed for all objects of the object class or object class stereotype defined with the XML attribute `ClassName`.

Defined XML Attributes	Rescan of Indicators performed for...
`ClassName` `Query` (child XML element)	all indicator types that are defined for the object class or for the object class stereotype defined with the XML attribute `ClassName` are re-computed for the subset of objects of the defined object class or object class stereotype found by the query.

`ClassName`
 	
`Evalua-tionType`
 	
`Query` (child XML element)
	all indicator types that belong to the evaluation type defined with the XML attribute ***EvaluationType*** will be re-computed for the subset of objects of the object class or object class stereotype defined with the XML attribute `ClassName` found by the query.

`ClassName`
 	
`Evalua-tionType`
 	
`Indica-torType`
 	
`Query` (child XML element)
	indicators with the indicator type and evaluation type defined with the XML attributes ***IndicatorType*** and ***EvaluationType*** will be re-computed for the subset of objects of the object class or object class stereotype defined with the XML attribute `ClassName` that are found by the query.

Running RescanIndicatorsConsole.exe

Executable	`RescanIndicatorsConsole.exe` located in the ***Programs*** subdirectory of the Alfabet installation directory
Remote or stand-alone access	Stand-alone access to the database with a server alias or remote access with a remote alias, connecting to a running Alfabet Server. ⓘ Verbose logging is only available when the tool is run in remote mode.
Precondi-tions	At least one evaluation type with defined indicator types is assigned to the object class that the batch calculation of indicators is executed for.
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see [Standard Logging for Alfabet Batch Utilities](#). **NOTE**: When the logging parameter `-logverbose` is selected, the log file may become extremely large. It is recommended that `-logverbose` is only used for trouble-shooting and not for productive runs. The parameter is only available if the tool is run in remote mode.

Command line help	Start executable with `-h` or `-help`

To execute the `RescanIndicatorsConsole.exe`, use one of the following command line statements depending on the functionality that you want to execute.

To rescan all indicators or all indicators of a defined evaluation type for an object class:

```
RescanIndicatorsConsole.exe-msalias <alias name>-alfaLoginName <user name>-
alfaLoginPassword <user password>-className <class name of Alfabet object
class> [-evaluationType <name of evaluation type>]
```

To rescan the indicators for an indicator type and a set of objects defined in an XML configuration file as described above:

```
RescanIndicatorsConsole.exe-msalias <alias name>-alfaLoginName <user name>-
alfaLoginPassword <user password>-configFile <path to and name of
configuration file>
```

The output of the batch calculation is by default directed to the command line but it can also be redirected to an output file.

The table below displays the command line options:

Command Line Option	Mandatory/ Optional	Explanation
`-className`	Optional (Mandatory if-`config-File` is not set)	Enter the name of the object class that the indicators will be calculated for. This option is ignored if-`configFile` is set. If-`configFile` is not set, this option is mandatory. You can either specify an object class or an object class stereotype. Object class stereotypes must be defined using the following syntax: *ObjectClassName*:*StereotypeName* In the **Class Configuration** functionality, evaluation types can be assigned to object classes or object class stereotypes. If the computed indicators are assigned to the object class, specify the object class. If the computed indicators are assigned to the object class stereotype, define the object class stereotype. If you define the object class while computed indicators are assigned to the object class stereotypes only, no re-calculation will be done. **NOTE**: The batch job can only be executed if the class exists in the Alfabet Meta-Model and an evaluation type with defined indicator types is assigned to the object class. The batch job is terminated with an error message in case the class specification does not match the conditions specified above.
`-evaluation-Type`	Optional	Enter the name of the evaluation that the indicators will be calculated for. This option is ignored if-`configFile` is set.

Command Line Option	Mandatory/ Optional	Explanation
		NOTE: When data translation is used, the name must be specified in the primary language.
`-configFile`	Optional (Mandatory if `-className` is not set)	Enter the name of the XML configuration file for the definition of the objects and indicators to be scanned and changed. The name specification can contain the path to the file either absolute or relative to the working directory of the batch utility. If `-configFile` is set, the command line options `-className` and `-evaluationType` are ignored. The information is read from the configuration file.
`-idletime`	Optional	The batch utility waits for alive messages from the Alfabet Server and if none are received within 60 seconds, the job is cancelled. The allowed wait time for alive messages can be changed by defining a new time interval in seconds with `-idletime`.
`-msalias <alias name>`	Mandatory	Enter the alias name as specified in the `AlfabetMS.xml` configuration file. If a remote alias is specified, the corresponding Alfabet Server must be running. If a server alias is specified, all other Alfabet applications with access to the Alfabet database must be stopped.
`-msaliasesfile <Alfabet configu­ration file path>`	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName <user name>`	Mandatory	User name for access to the Alfabet database. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPassword <user pass­word>`	Optional	Password for access to the Alfabet database.

Batch Evaluation of Color Rules with RescanColorRules.exe

A color rule is a set of query-based rules that either can be applied to a set of objects in the page views available for a map view or for diagram items in page views featuring Alfabet diagrams. Relevant objects found by the Alfabet queries will be displayed with the specified color. Once a color rule is defined on the **Color Rules** page view of a master plan and IT strategy in Alfabet, it must be activated in order for the

Alfabet query to be executed. The Alfabet query should be re-activated periodically in order to update the query results and include changes made to the database.

> ⓘ For more information about the definition of color rules, see the chapter *Configuring Color Rules for Map Views and Diagram Views* in the reference manual *Configuring Evaluation and Reference Data in Alfabet*.

The tool `RescanColorRules.exe` allows you to batch evaluate all color rules defined for master plans or IT strategies in Alfabet.

> ⓘ As an alternative to execution of `RescanColorRules.exe`, Alfabet users with access to the **Job Schedule** functionality can schedule the batch evaluation of color rules in defined intervals via a job schedule. For more information, see *Scheduling ADIF Jobs and Batch Jobs via the Job Schedule Functionality* in the reference manual *User and Solution Administration*.

Executable	`RescanColorRules.exe` located in the **Programs** subdirectory of the Alfabet installation directory
Remote or stand-alone access	Stand-alone access to the database with a server alias or remote access with a remote alias, connecting to a running Alfabet Server.
Preconditions	An `AlfabetMS.xml` configuration file with a server alias configuration specifying the connection to the Alfabet database is available and color rules are defined in Alfabet.
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see [Standard Logging for Alfabet Batch Utilities](#).
Command line help	Start executable with `-h` or `-help`

To execute the `RescanColorRules.exe`, use the following command line statement:

```
RescanColorRules.exe-msalias <alias name>-alfaLoginName <user name>-
alfaLoginPassword <user password>
```

The table below displays the command line options:

Command Line Option	Mandatory/ Optional	Explanation
`-msalias <alias name>`	Mandatory	Enter the alias name as specified in the `AlfabetMS.xml` configuration file. If a remote alias is specified, the corresponding Alfabet Server must be running. If a server alias is specified, all other Alfabet applications with access to the Alfabet database must be stopped.

Command Line Option	Mandatory/ Optional	Explanation
`-msaliasesfile` `<Alfabet configuration file path>`	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName`	Mandatory	User name for access to the Alfabet database. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPasswordBatch`	Optional	Password for access to the Alfabet database.

Batch Processes for Workflows with AlfaWorkflowCommandPrompt.exe

Alfabet supports your enterprise in defining and maintaining workflows in which you can track and coordinate the activities that should be performed by various persons in a particular sequence.

Software AG provides the tool `AlfaWorkflowCommandPrompt.exe` for batch processing of workflows.

Executable	`AlfaWorkflowCommandPrompt.exe` located in the ***Programs*** subdirectory of the Alfabet installation directory
Remote or stand-alone access	Stand-alone access to the database with a server alias or remote access with a remote alias, connecting to a running Alfabet Server.
Preconditions	An `AlfabetMS.xml` configuration file with a server alias configuration specifying the connection to the Alfabet database is available and Prior to batch execution for a workflow, a workflow template must be defined for the workflow with the tool Alfabet Expand. The definition of the workflow templates and workflow steps to execute the batch process is specified below. For general information on the configuration of workflows with workflow templates and an overview over the workflow for the execution of workflows see the chapter *Configuring Workflows* in the reference manual *Configuring Alfabet with Alfabet Expand*.
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see Standard Logging for Alfabet Batch Utilities.

| Command line help | Start executable with-`h` or-`help` |

When you start `AlfaWorkflowCommandPrompt.exe`, one of the following batch processes are performed depending on the batch job type specified for the call. Please note that none of the actions will be performed without running the batch job:

- **Sending reminder emails and setting the workflow step to Expired if a performance duration is configured for the workflow step**: A workflow step can be configured so that reminder emails can be automatically sent to the user responsible for the workflow step as well as to the workflow owner if there is no action taken on the workflow step for a configured period of time. This is configured by means of the **Performance Duration** attribute for the relevant workflow step. After the time period configured via the **Performance Duration** attribute has elapsed, the workflow step will be set to **Expired** and reminder emails sent out. Reminder emails will only be sent if this attribute is defined and the relevant email templates are available. (For more information, see the chapter *Configuring Workflows* in the reference manual *Configuring Alfabet with Alfabet Expand*.).

  ⚠️ The tool `AlfaWorkflowCommandPrompt.exe` must be executed regularly with the job type *check* in order to send reminder emails. It is recommended that you execute the workflow batch job on a daily basis as workflows may begin at any time which may require workflow reminder emails to be sent.

- **Automatic start of workflows for objects defined with a query**: A workflow template can be configured to allow for the automatic initiation of a workflow for all objects found in an Alfabet query. A workflow will be automatically started for each object found as a result of the query. To configure the automatic start of workflows for found objects, the **Automatic Start** attribute must be defined `True` for the relevant workflow template and a query must be defined for the **Base Objects via Query** attribute. Alternatively, you can restrict the execution of the batch process to workflow templates specified in the command line when starting `AlfaWorkflowCommandPromt.exe`. Please note that workflows cannot be created via batch process for workflow templates that have the **Workflow State** attribute set to `Plan` or `Retired`.

  If the **Automatic Start** attribute has been set to `True`, a workflow administrator can also execute the batch initiation of workflows via the **Start Automatically** button in the **Workflow Administration** functionality available via an administrative profile. For more information, see the section *Tracking and Managing Workflows* in the reference manual *User and Solution Administration*.

- **Automatic closure of workflow steps**

  A workflow step can be configured to be automatically closed and advance to the next workflow step if all post-conditions have been satisfied. Otherwise the workflow step must be manually closed by the responsible user in order to advance to the next workflow step. To configure the automatic closure of workflow steps, the **Allow Automatic Closure** attribute must be defined `True` for the relevant workflow step.

- **Automatic deletion of finished workflows**

  A workflow template can be configure to allow for the automatic deletion of all of its workflows that are in the state **Finished**. To configure the automatic deletion of finished workflows, the **Auto Delete** attribute must be defined `True` for the relevant workflow template.

- **Rescan and update of responsible users for current workflow steps in a workflow**

If the roles and responsibilities in your enterprise have changed, it may be necessary to update the Alfabet users who are responsible for a workflow. The batch utility can reassess the current workflow step of a selected workflow and update the assignment of responsible users to these workflow steps. Any new users found by the queries defined for the responsibility definition of the relevant workflow steps since the query was last executed will be sent the notification configured for the `OnEnterStep` workflow step action and the workflow step will be displayed in the **My Workflow Activities** view for all users found by the associated query.

- **Rescan and update workflow responsibilities**

    If the roles and responsibilities in your enterprise have changed, it may be necessary to update the Alfabet users who are responsible for a workflow. The batch utility reassesses the current workflow step of a selected workflow and update the assignment of responsible users to these workflow steps.

    The instructions associated with the responsibility definition of the workflow step as well as the query defined for the corresponding workflow step action of type Notification will be executed to find new responsible users.

    Any new users found by the query since its last execution will be sent the notification configured for the `OnEnterStep` workflow step action and the workflow step will be displayed in the **My Workflow Activities** view for all users found by the associated query.

Call up the `AlfaWorkflowCommandPrompt.exe` with the following command line:

```
AlfaWorkflowCommandPrompt.exe-<type of batch job> *|<workflow
name>{,<workflow name>}-msalias <alias name>-alfaLoginName <user name>-
alfaLoginPassword <user password>
```

The table below displays the command line options:

Command Line Option	Mandatory/Optional	Explanation
<type> <workflow name(s)>	Mandatory	Select one of the following types of batch job: • `Auto` to automatically start workflows • `Check` to send reminder emails and set workflow steps to **Expired**. • `StepClosure` to automatically close workflow steps • `DeleteFinished` to automatically delete finished workflows • `RescanResponsibility` to reassign workflow responsibilities The type of workflow is followed by the technical name of the workflow template(s) for which workflows are automatically generated (= **Name** attribute of the workflow template). If more than one workflow template is specified, the workflow template names must be specified in a comma-separated format without white spaces. If an asterisk (*****)

Command Line Option	Manda-tory/Op-tional	Explanation
		is specified, workflows are generated for all workflow templates for which the **Automatic Start** attribute is defined `True`.
`-msalias <alias name>`	Manda-tory	Enter the alias name as specified in the `AlfabetMS.xml` configuration file. If a remote alias is specified, the corresponding Alfabet Server must be running. If a server alias is specified, all other Alfabet applications with access to the Alfabet database must be stopped.
`-msaliasesfile < Alfabet configuration file path>`	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName`	Manda-tory	User name for access to the Alfabet database. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user. ⓘ If the workflow is started via the batch tool `AlfaWorkflow-CommandPrompt.exe`, then the workflow owner is based on the owner of the workflow template. If no workflow template owner is defined, then the current user executing the batch job is the workflow owner.
`-alfaLoginPass-word`	Optional	Password for access to the Alfabet database.
`-RescanMode <se-lectable mode>`	Optional	If the type of workflow is `RescanResponsibility`, three different rescan modes can be defined. • Select `ResetDelegation` to update the responsible users of all workflows steps including the workflow steps that have been reassigned to a different user through the step delegation action. Reassignments are not taken into account. • Select `ReapplyStepDelegation` to update the responsible users of all workflows steps and reapply any step delegation decisions users were previously performed through the step delegation action. In this case, responsible users may be added to or removed from a delegated workflow step. • Select `SkipDelegatedSteps` to update the responsible users of all workflows steps except the steps that have been

Command Line Option	Mandatory/Optional	Explanation
		reassigned to a different user through the step delegation action.

Batch Processes Relevant for the Alfabet Publication Framework

The Alfabet Publication Framework (APF) allows to publish current data from the Alfabet database to Microsoft® Word documents. Publication is done on basis of user defined Word templates including bookmarks that are ten mapped to relevant data of the Alfabet database via a publication configuration using the tool Alfabet Expand.

Two different methods for triggering publications based on existing publication configurations are available. Both include functionality that must be triggered by a command line utility.

- Publications can be triggered by a user logged in to the Alfabet user interface. To trigger a publication, the user must open a customer defined report that either allows to publish data about an object the user selects in the report or to publish data about the object the user currently works with when opening the report via a button in the toolbar.

  The publications defined by users via the Alfabet user interface are stored in the **Internal Document Selector** in the Alfabet database. The user can download the document either in the configured report or in a standard Alfabet functionality that lists all reports created during the last month. Documents that are older than one month are no longer displayed in the Alfabet user interface but are still available in the **Internal Document Selector**. Software AG provides a batch utility to delete old publications from the **Internal Document Selector**.. It is recommended the batch utility is run in regular intervals to clean the database from old publications.

  For more information about the batch utility deleting the old documents from the database, see Deleting Expired Publications from the Database.

- Publications can be triggered by a command line utility. The resulting zip file containing the published Microsoft® Word documents is stored directly in the local file system.

  For more information about the command line utility for publishing data in Microsoft® Word documents, see Triggering Publication via a Batch Utility.

ⓘ For general information about APF and the required configuration of publications see *Publishing Data In Microsoft® Word Format* in the reference manual *Configuring Alfabet with Alfabet Expand*.

Triggering Publication via a Batch Utility

Software AG provides a Windows® command line tool PublicationConsole.exe to publish data on basis of a configured publication to the local file system.

Executable	`PublicationConsole.exe` located in the **Programs** subdirectory of the Alfabet installation directory
Remote or stand-alone access	Remote access with a remote alias connecting to a running Alfabet Server.
Preconditions	An `AlfabetMS.xml` configuration file with a remote alias configuration specifying the connection to the running Alfabet Server is available and a publication definition suitable for execution via a batch job is defined and in the **State** `Active`.
Logging	Log information is written to a log file and to the Alfabet database. Logging is configurable in the command line. For more information about logging, see <u>Standard Logging for Alfabet Batch Utilities</u>.
Command line help	Start executable with `-h` or `-help`

Providing Relevant Data and Configuration

`PublicationConsole.exe` requires access to the following files and configurations:

A fully configured publication

A publication must be completely configured and set to the State `Active` before starting a publication job. The publication must be defined in the Alfabet database the Alfabet Server connects to and must be executable independent of selection of a base object.

An AlfabetMS.xml configuration file

For access to the Alfabet database, an `AlfabetMS.xml` configuration file with an alias configuration must be available. The `AlfabetMS.xml` configuration file must contain a remote alias configuration to access the Alfabet Server that is connected to the Alfabet database. When starting the publication console application, the alias configuration name must be specified in the command line. If the `AlfabetMS.xml` configuration file containing the alias configuration is not located in the working directory of the publication console application, the path to the file must additionally be specified in the command line.

Starting the Publication Console Application

Start `PublicationConsole.exe` with the command line options listed in the table below.

You must start `PublicationConsole.exe` with at least

- a specification of the alias configuration defining access to an Alfabet Server or a Alfabet database,

- a specification of the user name for log in to the Alfabet Server,

- the name of the publication and,

- the target location for the published files in the local file system:

```
PublicationConsole.exe-msalias <alias name>-alfaLoginName <user
name>-alfaLoginPassword <user password>-publication <publication
name>-outputfile <file name>.zip
```

The table below displays the command line options:

Command Line Option	Mandatory/ Default	Explanation
`-msalias` <alias name>	Mandatory	Enter the remote alias name as specified in the `AlfabetMS.xml` configuration file. The corresponding Alfabet Server must be running.
`-msaliasesfile` < Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName` <user name>	Mandatory	User name for access to the Alfabet database. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPassword` <user password>	Optional	Password for access to the Alfabet database.
`-publication` <publi­cation name>	Optional	Specify the name of the publication defined in Alfabet Expand.
`-outputfile` <file name>	Mandatory	Specify the name of and path to the ZIP file containing the publication output.

Command Line Option	Manda-tory/ Default	Explanation
		NOTE: The extension of the file name must be.zip. The file with the defined name is created during the publication process and contains the published word document(s).
-singledocument <true/false>	Optional	this option is only relevant when data about multiple objects is published. When-singledocument is set to true, data about all selected objects is published in one document. Otherwise, one document is published per selected base object.

-<parameter name> <parameter value>	Optional	When the queries in the publication definition are configured with Alfabet query language parameters, the parameter can be specified in the command line. For each variable, a separate command line option must be specified as:
		
		    -<parameter name> <value>
		
		for example:
		
		    -fiscalyear '2009'
		
		For more information about the use of variables, see Batch Processes Relevant for the Alfabet Publication Framework.

Defining Publication Output in the Command Line of the Batch Utility

When a publication is triggered by PublicationConsole.exe, the selection of the base objects of the publication must be defined via a query in the publication definition. This results in a publication returning results for the same set of base objects each time it is executed.

If you want to select different base objects for each publication with PublicationConsole.exe, you can use Alfabet query language parameters in the queries selecting the base object as well as in the queries defining the data to be published about the objects. The value of the parameter is then defined in the command line of the publication console application when triggering a publication.

For more information about how to define parameters in Alfabet queries, see *Referring to the Current Alfabet Context in a WHERE Condition* in the chapter *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*.

For more information about how to define parameters in native SQL queries, see *Using Alfabet Parameters* in the section *Defining Native SQL Queries* of the chapter *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*.

The following configuration is required to use parameters in the command line:

- When defining the publication, define queries that contain Alfabet query language parameters in WHERE conditions.

- Optionally, define a default value for each parameter in the **Debug Arguments** attribute of the publication. The definition must be defined in the following syntax:

  - Each default must be defined as <parameter name>=<value>.

  - When multiple default values are added, they must be defined comma-separated.

  - The <parameter name> does not include the prefix @ or: of the Alfabet query language parameter.

  - The <value> must be written in the attribute as if defined in a query. For example, if a string is written in inverted commas in the query, the inverted commas will be part of the value specification.

- When starting the command line application, the value substituting the Alfabet query language parameter must be defined in the command line as-<parameter name><value>. The parameter name is the Alfabet query language parameter without the prefix @ or: and the value must be defined as required by the query syntax. This means that a string written in inverted commas in the query must be defined with the inverted commas.

A publication is defined that triggers the publication of data about a number of applications found via their name. The query finding the application is defined in the publication entry as:

```
ALFABET_QUERY_500

FIND Application

WHERE Application.Name LIKE:AppName
```

The attribute **Debug Arguments** of the publication definition specifies a default value for the application name. When no parameter value is defined in the command line, the publication will be created about all applications with a name starting with CMS:

```
AppName='CMS%'
```

When starting `PublicationConsole.exe` to create the publication about an application named "Central Server", the command line must be specified as follows:

```
PublicationConsole.exe-msalias Production-alfaLoginName SystemAdmin-
alfaLoginPassword AdminPassword-publication ApplicationData-
outputfile CentralServer.zip-AppName 'Central Server'
```

Deleting Expired Publications from the Database

Software AG provides a Windows® command line tool `AlfaBatchExecutor.exe` to delete expired publications that are no longer displayed in the user interface from the **Internal Document Selector** of the Alfabet database.

| Executable | `AlfaBatchExecutor.exe` located in the ***Programs*** subdirectory of the Alfabet installation directory |

Remote or stand-alone access	Stand-alone access to the database with a server alias. Remote access with a remote alias connecting to a running Alfabet Server is only possible for the batch processing of monitors.
Preconditions	An `AlfabetMS.xml` configuration file with a server alias configuration specifying the connection to the Alfabet database is available and the respective functionality is configured.
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see Standard Logging for Alfabet Batch Utilities.
Command line help	Start executable with `-h` or `-help`

The executable must be started with the following parameters:

```
AlfaBatchExecutor.exe-jobClass ExpiredReportDeletionJob-msalias <alias
name>-alfaLoginName <user name>-alfaLoginPassword <user password>
```

The table below displays the command line options:

Command Line Option	Manda-tory/Op-tional	Explanation
`-jobClass` <job name>	Mandatory	Enter `ExpiredReportDeletionJob`.
`-msalias` <alias name>	Mandatory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file. All other Alfabet applications with access to the Alfabet database must be stopped.
`-msaliasesfile` < Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName`	Mandatory	User name for access to the Alfabet database. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPassword`	Optional	Password for access to the Alfabet database.

Chapter 7:   Routine Operational and Maintenance Tasks

This chapter addresses regular tasks that the system administrator must perform to maintain the system including, for example, database backups and updates.

The following information is available:

-
  - Planned Outages of the Alfabet Components
    - General Workflow for Planned Shutdowns
    - Shutting Down the Alfabet Application
    - Starting the Alfabet Application
- Unplanned Shutdowns of the Alfabet Components
  - Monitoring the Availability of Alfabet Components
    - Monitoring the Alfabet Server Service
    - Monitoring the Alfabet Web Application
    - Return Code
  - Releasing the Restricted Mode
- Logging of Alfabet Functionality
  - Central Logging of Functionality for Alfabet Components
  - Checking the Accessibility of the Alfabet Database
    - Triggering Monitoring Event Execution via the Alfabet User Interface
    - Triggering Monitoring Event Execution via a Command Line
- Carrying Out Database Maintenance Tasks
  - Backing Up and Recovering the Alfabet database
  - Rebuilding Defragmented Indices
    - Index Defragmentation on Server Start
    - Index Defragmentation via a Command Line Job
    - Index Defragmentation During ADIF Import and Event Execution
  - Administration Tasks Related to Solution Design
    - About the Storage of the Alfabet Meta-Model in the Alfabet database
    - Activating Update of ALFA_MM_*_INFO and ALFA_PM_*_INFO Tables During Update of the Meta-Model via AMM File
    - Storing the Configuration of a Database in an AMM Update File
    - Restoring the Configuration of the Alfabet Solution Environment

Planned Outages of the Alfabet Components

Planned outages of the system are necessary, for example, to perform upgrades or to maintain the Web server or the database server. In addition, some configuration tasks require a restart of the Web server hosting the Alfabet Web Application to become visible to the users.

Typical tasks requiring a planned outage and restart of one or multiple system components and the required action that must be taken for the different Alfabet components are the following:

Upgrade to a new Alfabet release:

- **Database Server**: During upgrades the database server must run. The meta-model changes for the release are implemented in the database during upgrade.

  Please note that database replication mechanisms must also be shut down during upgrades.

- **Web Server**: During upgrades, the connection between the Alfabet database and the Alfabet Web Application is closed and the files in the working directory of the Alfabet Web Application are overwritten. You can configure the Alfabet Web Application to display an outage message during the upgrade process to inform users about the outage. The Web Server needs to be re-started at the end of the upgrade process.

- **Component for Administration and Configuration**: During an upgrade, all Alfabet components must be shut down.

Maintenance of the Web Server

- **Database Server**: The database server is not affected by this action.

- **Web Server**: Clients cannot access the Alfabet Web Application and should be informed in advance (for example, via broadcast messages or via email about the outage).

- **Component for Administration and Configuration**: The following Alfabet components are part of the Alfabet Web Application and cannot be used as long as the Web server is not running:

  - Alfabet Expand Web including the Guide Pages Designer

- all applications using the Alfabet RESTful API to access the Alfabet database including the RESTful services based ARIS- Alfabet Interoperability Interface

- all applications using the Alfabet Web services including the Web services based ARIS- Alfabet Interoperability Interface (the Alfabet Web Services are implemented separate from the Alfabet Web Application. If they are implemented on a different Web Server (host), they might not be affected).

All other Alfabet components can still work and connect to the database while the web application is not available.

Maintenance of the Database Server

- **Database Server**: The database server should only be stopped after shutting down all Alfabet components.

- **Web Server**: The Alfabet Web Application should be configured to display an outage message before shutting down the database server.

- **Component for Administration and Configuration**: All Alfabet components must be shut down before shutting down the database server.

Updating the Alfabet meta-model or the whole database content with the update meta-model and restore database mechanisms of the Alfabet Administrator or changing the class model with Alfabet Expand:

- **Database Server**: During these actions, all other connections to the Alfabet database are closed and are only re-established after the changes are implemented in the database.

  Please note that database replication mechanisms must also be shut down during updates of the meta-model.

- **Web Server**: The Alfabet Web Application should be configured to display an outage message during the process.

- **Component for Administration and Configuration**: The Alfabet components cannot connect to the Alfabet database and should be shut down before changing the meta-model.

Updating the Alfabet configuration objects of the meta-model with Alfabet Expand:

- **Database Server**: Meta-model changes affecting the configuration objects like object profiles, page views or class settings can be stored in the Alfabet database during runtime of the Alfabet components. All current connections to the database remain open.

- **Web Server**: The Web Server needs to be re-started to make the changes available to the Alfabet users.

- **Component for Administration and Configuration**: The Alfabet Server needs to be re-started to make the changes available to client processes.

  Alfabet Expand does not need to be restarted. Meta-model changes can be read to the application via a menu button interaction.

General Workflow for Planned Shutdowns

The following workflow for the shutdown and start up of the Alfabet components is relevant for all planned outages. The procedures are described in detail in the following sections.

Keep the following principles in mind for planned shutdowns:

- The Alfabet components must be shut down via a top-down procedure. In other words, you must first shut down the Alfabet Web Application before shutting down the database server. If the Alfabet Web Application is configured to work with an Alfabet Server you must first shut down the Alfabet Web Application before shutting down the Alfabet Server.

- The Alfabet components must be started via a bottom-up procedure. In other words, you must first start the database server before starting the Alfabet Web Application.

- Depending on the task you want to carry out, you may not need to shut down or start up all involved components. For example, if you plan maintenance work on the web server host, and the database server runs on a separate host, you will not need to shut down the database server.

- For many tasks, the database server must be actively running whereas the Alfabet Web Application must be shut down. As a general rule, you should only shut down the Alfabet components down to the component that must be inactive in order to carry out your task.

- In the following procedure, the database server remains active. Shutdown of the database server is normally not required for maintenance tasks. If you want to shut down the database server, the database server must be the last component to shut down and the first to be started during start-up. Please note that database replication mechanisms must also be shut down during upgrades.

- To upgrade to a new patch or release of Alfabet, the database server must be active in order to carry out the upgrade process. Please note that database replication mechanisms must also be shut down during upgrades.

- In a production environment, the Alfabet Server is normally run as a service. The Alfabet Server can also be started during a user session as an application (alfaServer.exe). This mode is for ad-hoc and test operation only. If the Alfabet Server is operated manually, use the functionality **File** > **Shutdown** in the menu of the Alfabet Server to shut down the Alfabet Server and use the functionality **File** > **Start** to restart it.

> ⚠ Do not use the functionality **File** > **Restart**. This functionality is designed for special purposes only and is not supported for normal restart in planned or unplanned outages.

In the following procedure, the Alfabet Web Application is not shut down during the planned outage, but rather is configured to display a message to inform users about the system shutdown.

Procedure for complete shutdown:

1) Configure the Alfabet Web Application to display a system outage message instead of opening the Alfabet interface.

2) If applicable, disable all scheduled tasks and replication mechanisms concerning the system.

3) If applicable, shut down the Alfabet Server Service.

4) If applicable, shut down the database server.

Procedure for start-up:

1) If applicable, start the database server.

2) If applicable, start the Alfabet Server Service.

3) Configure the Alfabet Web Application to access Alfabet instead of displaying the outage message.

4) Reset the application pool of the Alfabet Web Application.

5) Test the availability and functionality with the Alfabet Web Client s

6) If applicable, enable all scheduled tasks concerning the system.

Shutting Down the Alfabet Application

The procedure described here is for shutdown of a production environment with a running Alfabet Server and a running Alfabet Web Application.

The following steps are required to shutdown the Alfabet components:

- Step 1: Configuring the Alfabet Web Application to Display a System Shutdown Message

- Step 2: Shutting Down Scheduled Tasks and Database Replication Processes

- Step 3: Shutting Down the Alfabet Server Service

Step 1: Configuring the Alfabet Web Application to Display a System Shutdown Message

This step is an optional but useful mechanism to inform users about current maintenance work on the Alfabet system and prevent complaints about an unexpected server shutdown.

During a planned system outage, you can temporarily substitute the standard HTML start page with another file by means of the Internet Information Services® Manager on the Web server that hosts the Alfabet Web Application. The standard HTML start page to access Alfabet via the Alfabet Web Application is the Home.aspx file located directly in the sub-directory of the Alfabet Web Application, which is located in the root directory of the Alfabet installation.

> You can also configure a broadcast message to inform all system users who log in to the Alfabet Web Application about the upcoming planned shutdown. All broadcast messages that have been activated will be displayed in the banner at the bottom of the Alfabet screen. Broadcast messages are configured in the **Broadcast Messages** functionality available in the Alfabet user interface via an administrative user profile. For more information, see the section *Defining Broadcast Messages for the User Community* in the reference manual *User and Solution Administration*.

The following information is relevant for this procedure:

Number	Parameter	Value
13	Web Server Host	

Number	Parameter	Value
14	Application directory for the Alfabet Web Application	
16	Path to the physical directory of the Alfabet Web Application	
17	Path to the HTML page displaying the outage message	

To substitute the standard start file with another file providing inforamtion about the current system outage, do the following in the Internet Information Services® Manager on the Web Server host (`13`):

In Internet Information Services® 8:

1) Create an HTML page displaying the outage message (`17`) and store it in the sub-directory of the Alfabet Web Application in the root directory of the Alfabet installation (`16`).

2) In the explorer tree of the Internet Information Services® Manager, locate the application directory of the Alfabet Web Application (`14`).

3) In the **Features View**, double-click **Default Document** in the section **IIS**.

4) In the **Default Document** page, remove the standard document `Home.aspx` and any other configured standard documents.

5) In the **Actions** pane, click **Add** and enter the name of the HTML file containing the system shutdown message.

6) Close the Internet Information Services® Manager.

Step 2: Shutting Down Scheduled Tasks and Database Replication Processes

Software AG provides several batch utilities that must be executed to activate specific Alfabet functionalities, to enhance the usability of some Alfabet functionalities, or to update data in the Alfabet database.

Executables for batch jobs can be started in a command line or by means of a Windows® batch job. When executing a Windows batch job, the execution time for a batch job is defined with the help of the Windows scheduler for batch jobs.

Prior to shutting down the Alfabet Server Service, all Windows® batch jobs running Alfabet batch utilities that connect to the Alfabet Server must be set to disabled.

Executables for batch jobs and ADIF import and export can be scheduled for regular execution via the Job Schedule functionality on the Alfabet user interface. To avoid interruption of scheduled jobs during maintenance, job schedules can be configured to read information about periods blocked for maintenance work from an XML definition. Any jobs scheduled for start during the maintenance period will then be postponed to start after the end of the maintenance period. If the Job Schedule functionality is used, information about the planned maintenance outage should be provided via the XML object ***MaintenanceWindows***. For more information, see Defining Maintenance Windows for Scheduled Jobs below.

Prior to updating the meta-model, any database replication processes that are applied to the Alfabet database must be shut down.

Step 3: Shutting Down the Alfabet Server Service

To shut down the Alfabet Server service:

1) On the Web Server host, click the **Start** icon that appears when you move the mouse to the lower left corner and click on the **Server Manager** icon to open the Server Manager.

2) In the menu on the upper right of the Server Manager, select **Tools** > **Services**.

3) In the Services window, select the Alfabet Server service in the list of services and click **Stop the service** on the left of the list.

Starting the Alfabet Application

The following steps must be executed in the sequence described below in order to start the Alfabet application:

> ⓘ If the database server has been shut down, it must be started prior to execution of the following steps.

- Step 1: Starting the Alfabet Server Service

- Step 2: Reconfiguring the Alfabet Web Application to Access the Alfabet Database

- Step 3: Testing Connectivity and Functionality

- Step 4: Enabling Scheduled Tasks and Database Replication Mechanisms

Step 1: Starting the Alfabet Server Service

1) On the Web Server host, click the **Start** icon that appears when you move the mouse to the lower left corner and click on the **Server Manager** icon to open the Server Manager.

2) In the menu on the upper right of the Server Manager, select **Tools** > **Services**.

3) In the Services window, select the Alfabet Server service in the list of services and click **Stop the service** on the left of the list.

> To start the Alfabet Server Service with each restart of the operating system, right-click the Alfabet Server Service in the list of services and select **Properties**. In the **General** tab of the editor that opens, set the **Startup Type** to **Automatic (Delayed Start)**.
>
> If you are not a local administrator, open the **Log On** tab and change the **Log on as:** option to **This account:**. In the **This account:** field specify the account of the domain administrator with <domain>\<administrator> and enter the account's password in the **Password** and **Confirm Password** fields and click **Apply**:

Step 2: Reconfiguring the Alfabet Web Application to Access the Alfabet Database

If you redirected the standard start page for the Alfabet Web Application to another HTML page with a shutdown message, you must now reset the original `Home.aspx` or `index.html` file of the Alfabet Web Application:

The following information is relevant for this procedure:

Number	Parameter	Value
13	Web Server Host	
14	Application Directory for the Alfabet Web Application	
15	Application Pool used for the Alfabet Web Application	

To substitute the standard start file with another file providing information about the current system outage, do the following in the Internet Information Services® Manager on the Web Server host (13):

In Internet Information Services® 8:

1) In the explorer tree of the Internet Information Services® Manager, locate the application directory of the Alfabet Web Application (14).

2) In the **Features View**, double-click **Default Document** in the section **IIS**.

3) In the Default Document page, remove the standard document displaying the outage message.

4) In the **Actions** pane, click **Add** and enter `Home.aspx` or `index.html`.

5) In the explorer tree of the Internet Information Services® Manager, locate the application pool of the Alfabet Web Application (15).

6) Reset the application pool.

7) Close the Internet Information Services® Manager.

Step 3: Testing Connectivity and Functionality

You can test the connectivity of the Alfabet Clients as described below. After testing the connectivity, it is recommended that you test the performance with functionality tests that are typical for your enterprise's use of Alfabet.

1) Open the Browser.

2) Enter the URL of the Alfabet Web Application (for example: http://<full server name>/ Alfabet).

3) Click **OK** to connect.

4) Login to Alfabet.

5) Select a user profile. The Alfabet user interface opens.

Step 4: Enabling Scheduled Tasks and Database Replication Mechanisms

After successful update, set all Windows® batch jobs running Alfabet batch utilities to enabled.

Database Replication Mechanisms can be re-started.

Unplanned Shutdowns of the Alfabet Components

An unplanned outage can occur, for example, because the server host has failed, an electrical outage has occurred, or the database server is not available. In any case, the system should be thoroughly tested before restarting the Alfabet components.

To restart the Alfabet components after an unplanned shutdown, see the section Starting the Alfabet Application.

The following mechanisms can be helpful to handle unplanned shutdowns:

- Monitoring the Availability of Alfabet Components

-

-

-

-

Monitoring the Availability of Alfabet Components

Software AG provides a batch utility that allows the Alfabet Server Service and the Alfabet Web Application to be monitored. The batch utility checks the availability of the Alfabet Web Application or the Alfabet Server Service in regular intervals and returns a status code that can be used to trigger an alert in case an unplanned shutdown occurs.

Executable	`AlfaServiceMonitorConsole.exe` located in the ***Programs*** subdirectory of the Alfabet installation directory
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see Standard Logging for Alfabet Batch Utilities.
Command line help	Start executable with `-h` or `-help`

The command line utility can be used to check the availability of single Alfabet components or to check the availability and database connectivity of all Alfabet components for which a connection is defined in a log file simultaneously. If all components are checked simultaneously and the central logging of the Alfabet components is configured to write into the Windows® Event Log with an event ID, the log information will not only be returned via the command line, but also written into the Windows Event Log.

Monitoring the Alfabet Server Service

The executable must be started with the following parameters:

```
AlfaServiceMonitorConsole.exe-Alfabetserverhost <host name>-
Alfabetserverport <port number>-Alfabetserver <server name>
```

The table below displays the command line options:

Command Line Option	Mandatory/ Optional	Explanation
`-Alfabetserverhost <host name>`	Mandatory	Enter the server host name for connections to the Alfabet Server as specified in the server alias configuration of the Alfabet Server with the **Host** attribute.

Command Line Option	Mandatory/ Optional	Explanation
`-Alfabetserver-port <port number>`	Mandatory	Enter the server port for connections to the Alfabet Server as specified in the server alias configuration of the Alfabet Server with the **Port** attribute
`-Alfabetserver <server name>`	Mandatory	Enter the name of the Alfabet Server as specified in the server alias configuration of the Alfabet Server with the **Server** attribute.

Monitoring the Alfabet Web Application

The executable must be started with the following parameters:

```
AlfaServiceMonitorConsole.exe-webserverhost <host name>-webserverport <port
number>-webserverpath <path>
```

The table below displays the command line options:

Command Line Option	Mandatory/ Optional	Explanation
`-webserver-host <host name>`	Mandatory	Enter the name of the Web server host.
`-webserver-port <port number>`	Optional	Enter the port for connections to the Web server.

`-webserver-path <path>`	Optional	Enter the path to the Alfabet Web Application on the server host. For example, if the Alfabet Web Application has the URL:
		
		`http://pITWebServer/Alfabet`
		
		The resulting path is:
		
		`Alfabet`

Return Code

Command Line Option	Mandatory/ Optional	Explanation
`-webserverhost <host name>`	Mandatory	Enter the name of the Web server host.

The server returns the following status code to inform about the availability of the Alfabet components:

Return code	Status
0	OK
1	Error
5500	Unable to find configuration file
5501	The definition of the connection to the Web Server is incorrect.
5502	The Web Server port could not be opened.
5503	The web site of the Alfabet Web Application is not accessible.
5504	The definition of the connection to the Alfabet Server is incorrect.
5505	The port on the Alfabet Server host could not be opened.
5506	A connection to the Alfabet Server could not be established.
5507	The connection between the Alfabet Server and the Alfabet database could not be established.
5508	The definition of the connection to the Alfabet database is incorrect.
5509	A connection to the Alfabet database could not be established.

Return code	Status
5510	The JSON file has an unvalid parameter definition.

Releasing the Restricted Mode

During critical operations such as to update or restore the Alfabet database, a restricted mode that inhibits any other than the current user to access the Alfabet database is set for the connection to the Alfabet database.

The restricted mode automatically ends with the end of the critical operation. However, it is possible that the restricted mode may persist even if the critical operation has ended. In this case, you will need to release the restricted mode for the relevant database using the Alfabet Administrator.

To release the restricted mode:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer and right-click a server alias connecting to the Alfabet database for which you want to release the restricted mode and select **Release Restricted Mode**.

2) A warning is displayed because the restricted mode may also be caused by an active meta-model update process. Make sure that no update or restore process is currently performed and then select **Yes** to release the restricted mode.

> To release the restricted mode of a Alfabet database stored on a Microsoft SQL Server®, the Alfabet SQL login account needs to be a member of the sysadmin or process admin server role.

If the manual release of the restricted mode is not adequate to release the restricted lock in case of unplanned outages (for example, because the restart of components is done automatically without user interaction), you can alternatively release the restricted mode via a command line tool provided by Software AG:

Executable	`AlfaAdministratorConsole.exe` located in the **_Programs_** subdirectory of the Alfabet installation directory
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see Standard Logging for Alfabet Batch Utilities.
Command line help	Start executable with `-h` or `-help`

The executable must be started with the following parameters:

```
AlfaAdministratorConsole.exe-msalias <alias name>-DbUser <username>-
DbPassword <password>-db_releaserestrictedmode
```

The table below displays the command line options:

Command Line Option	Mandatory/ Default	Explanation
`-db_re-leaserestricted-mode`	Mandatory	To release the restricted mode, start the console application with `-db_releaserestrictedmode`
`-msalias <alias name>`	Mandatory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.
`-msaliasesfile <Alfabet configuration file path>`	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-DbUser <user name>`	Mandatory	User name for access to the Alfabet database on the database server. ⓘ If the access mode to the Alfabet database (27) is Windows Authentication, the command line option `-DbUser` must be defined without a value for the user name and `-DbPassword` is not required.
`-DbPassword <user password>`	Optional	Password for access to the Alfabet database on the database server.

Logging of Alfabet Functionality

The following logging capabilities are available to ease control of the functionality of the Alfabet components:

- [Central Logging of Functionality for Alfabet Components](#)

- [Checking the Accessibility of the Alfabet Database](#)

  - [Triggering Monitoring Event Execution via the Alfabet User Interface](#)

  - [Triggering Monitoring Event Execution via a Command Line](#)

Central Logging of Functionality for Alfabet Components

The Alfabet components can be configured to write error messages to a defined log file on the local file system or to the Windows Event Log or to sent them to a Seq® server. The log message processing can be defined separately for different kind of log messages.

> (i) Please note the following:
>
> - Alfabet Batch Utilities write log files to separate log files. Execution of batch utilities is not included to central logging.
>
> - Sending of emails, and the execution of ADIF jobs, events and scheduled jobs can be monitored directly on the Alfabet user interface. For more information, see the reference manual *User and Solution Administration*.

Central logging is defined in the server alias of the Alfabet component:

1) Open the Alfabet Administrator.

2) Click the **Alfabet Aliases** node in the **Administrator** explorer. The right pane displays a list of all available alias configurations.

3) Click the server alias in the list and click the **Edit** button.

4) In the editor that opens, go to the **Server Settings** > **Logging** tab.

5) Activate the **Activate Logging** checkbox.

6) Define the location of the log information:

   - Log information can be written to a log file in the local file system. The log file name is fixed and has the following syntax: `Alfabet_<release number>_<component>_<server alias name>.log`. A separate log file is written for each Alfabet component. The log file will be archived, and a new log file started if either a defined time has elapsed, or a defined file size has been reached. The archived log files are each stored in a ZIP file. Both ZIP file and archived log file name are identical to the original log file name amended with either a timestamp or a GUID.

     > ⚠ Please note that only one archive ZIP file is created for each log file and that the ZIP file will be overwritten each time that log file content is archived. Zip archives must

be renamed or moved to another location by external mechanisms to be archived persistently.

To enable writing of log information into log files, define the file location and handling in the **File Logging** box.

- **Log File Directory**: Enter the absolute path of the log file directory or select a directory from the local file system using the Browse ... button. Both the log files and the archive ZIP files for the log files will be written into the selected directory.

  > ⓘ Server variables can be used to define part of the information as a variable with the value set in the **Variables** tab of the server alias configuration. For more information, see Defining Connections Based On Server Variables.

- **Log File Max Part Size MByte**: Define the maximum allowed file size in megabyte. The log file will be archived, and a new log file started if either the defined file size has been reached or the time defined with the **Log Rotation Max. Time (Hours)** attribute has elapsed.

- **Log Rotation File Name Suffix**: Select GUID to add a GUID to the ZIP file and the log file in the ZIP file on archiving of the log file.

- **Log Rotation Max Time (Hours)**: Define the maximum time between archiving of the log file in hours. The log file will be archived, and a new log file started if either the defined time has elapsed or the file size defined with the new **Log File Max Part Size MByte** has been reached.

- If information shall be written to a Seq server, the information will be provided to the Seq server via its HTTP API with a valid access key. The access information for the Seq server must be entered in the following fields in the **External Logging** box:

  - **Server Type**: Currently only Seq can be selected.

  - **Server URL**: The URL of the HTTP API of the Seq server.

  - **Access Key**: The access key for access to the HTTP API of the Seq server.

  - **Environment**: If multiple instances are run on the same host for production, validation and development and all instances are configured to send log messages to the same Seq server, the type of environment can be selected from the drop-down list. The selected environment type will be included in the messages to allow the different environments to be distinguished.

- If information shall be written to the Windows® Event Log, you can specify the following in the **Windows Event Logging** box:

  - **Event Source Name**: Optionally alter the name that is used as **Source** in the Windows Event Log. By default the **Event Source Name** parameter is the server alias name.

  - **Event Base ID**: If this attribute is defined, an event ID will be written into all entries in the Windows Event Log. The event ID is calculated from the event base ID defined in this attribute plus the logging level number. The logging level number is 2 for `Error`, 4 for `Warning`, 8 for `Information` and 16 for `Debug`. The event base ID must be an integer between 30000 and 65535.

7) Go to the **Server Settings** > **Logging Details** tab.

8) The log levels can be fine-tuned individually for different processes such as Alfabet RESTful services, ADIF execution, or event management. The table lists the following logging details for separate activation:

- ***System***: Activates logging of the basic functionalities of the Alfabet component. In addition, the following information is logged:

    - Information about the execution of jobs scheduled in the **Job Schedule** functionality is written to the log file if the log level **Information** is selected.

    - Information about JSON responses for ADIF sessions executed in the scope of the ARIS-Alfabet Interoperability Interface are written to the log file if the log level **Debug** is selected.

- ***UserLogon***: Activates writing of the following information to the Windows® Event Log: the registration in the event log, shutdown of the Alfabet Web Application and the reason for shutdown, and user log in and log out. If user authentication and authorization is done via synchronization with an external LDAP server, each LDAP authentication, LDAP authorization and search for a user in LDAP is also logged. The respective messages all start with LDAP.

    > ⓘ Please note the following:
    >
    > - Logging to file or Seq server is not supported for this log subject.
    >
    > - If multiple Alfabet components are involved in a login action and both are configured to write messages to the Windows Event Log, both components will write a separate message for the action. Management of authentication via data synchronization with an external LDAP server may also result in multiple entries per login.
    >
    > - When a user logs in or out of the Alfabet components, the user name and the information whether login or logout was successful will be logged. This does not only include login/logout to the Alfabet user interface but all processes that require a login such as sending requests to the Alfabet RESTful API.
    >
    > - If the **Server Settings** > **General** > **Update History User Name** attribute of the server alias is set to TECH_NAME, the **Technical Name** defined for the user will be used in the Windows Event Log instead of the user's name. It is essential to define the **Technical Name** attribute for all users if this setting is used. The **Technical Name** can only be used in the Windows Event Log if authentication is performed via the user data in the Alfabet database.

- ***Workflow***: Activates logging for the Alfabet workflow functionality.

- ***ADIF***: Activates logging of ADIF execution. The log information written to the log file for ADIF processes includes information about commit actions as information and about rollback actions as warnings. To include logging for the Job Schedule functionality, the ***ADIF*** log level must be Debug.

- ***Report***: Activates logging for the execution of configured reports.

- ***REST***: Activates logging of the Alfabet RESTful API functionality. If the Alfabet RESTful services are used, it is recommended to activate central logging of REST functionalities because the return values for the RESTful services do not cover all error details for security reasons. For example, if the Alfabet Server is required to execute an ADIF job that shall be

executed via a RESTful service call, the RESTful services will inform about forbidden access only and the information about the component that could not be accessed is only provided via central logging.

- **SSO**: Activates logging of activity related to a configured SAML authentication. This setting is only relevant if the Alfabet Web Application is configured to use SAML for user authentication.

- **Event**: Activates logging of the execution of events configured via the **Event Management** functionality.

  > (i) Log messages for event execution is also written to the `ALFA_EVENT_BUS` table in the Alfabet database.

- **Conditions**: Activates logging for the execution of conditions. Conditions are defined in Alfabet Expand to hinge visibility of elements of the Alfabet user interface on the availability of defined preconditions.

- **License**: Activates logging for the Alfabet license manager functionality.

- **MonitoringEvents**: Activates logging for the monitoring of database availability. This option must be activated to test the database availability via the mechanism described in the section Checking the Accessibility of the Alfabet Database below.

- **WebApplication**: Activates logging of web-based functions of the Alfabet Web Application. Please note that activation of this log option is increasing the number of messages written to the log file considerably. It is recommended to activate this option only if the cause of issues cannot be detected via the logging provided by the other log options.

- `Email`: Activates logging of automatic sending of emails.

9) Set the following in the rows of all the logging details that you would like to activate:

- In the **Level** column, define the log level. The following log levels are available:

  - `None`: no log output.

  - `Error`: only messages about errors are logged.

  - `Warning`: error messages and warning messages are logged.

  - `Information`: error messages, warning messages, and informational messages are logged.

  - `Debug`: error messages, warning messages, informational messages and all debugging and tracing messages are logged.

    > (i) For the options **Workflow**, **Report** and **Conditions**, only error and warning messages are available. The log levels `Information` and `Debug` should not be selected for these options.

- Click in the cell in the **File** column to activate writing of the log messages to the central log file on the local file system defined in the **Server Settings** > **Logging** tab with the **File Logging** setting. An **x** will be displayed in the cell after activation.

- Click in the cell in the **External** column to activate writing of the log messages to the external log server defined in the **Server Settings** > **Logging** tab with the **External Logging** settings. An **x** will be displayed in the cell after activation.

- Click in the cell in the **Event** column to activate writing of the log messages to the Windows® Event Log with the component name defined in the **Server Settings** > **Logging** tab with the **Event Source Name** attribute. An **x** will be displayed in the cell after activation.

10) Click **OK** to save your changes.

11) If you configured the central logging to write information to the Windows Event Log, do the following:

1) In the explorer of the Alfabet Administrator, right-click the server alias and select **Register Event Logger** in the context menu.

2) Click **OK** to save your changes and close the Alfabet Administrator.

3) Open the Windows Event Log and check whether the registration is logged for the specified **Event Source Name**.

> The registration of the Event Logger must be re-executed with the Alfabet Administrator run as administrator after each change to the configuration described above.

Checking the Accessibility of the Alfabet Database

A mechanism has been implemented that checks whether reading data from the Alfabet database and writing data to the Alfabet database is currently possible without causing any issues. The test is executed via a monitoring event that creates a test object, reads data about the test object, updates it, and deletes it. If errors occur during one of the processes, these will be logged.

The monitoring event can either be run once or scheduled to be executed in regular intervals of ten minutes.

The following preconditions are required for execution of the monitoring event and logging of the results:

- The one-time execution of the event requires a running database server hosting the Alfabet database, and a running Alfabet Web Application connected to the Alfabet database.

- The cyclic execution of the event requires a running database server hosting the Alfabet database, a running Alfabet Web Application connected to the Alfabet database., and a running Alfabet Server connected to the Alfabet database.

  The RESTful services of the Alfabet Web Application must be implemented, and a user must be selected for execution of self-reflective events. The API access **Has MonitoringAPI Access** option must be activated for that user and in the server alias of the Alfabet Web Application.

  > For more information about the implementation of the Alfabet RESTful services, see the reference manual *Alfabet RESTful API*.

- Central logging is enabled in the **Server Settings** > **Logging** tab of the server alias and the `Monitoring` log option is activated in the table of the **Server Settings** > **Logging Details** tab.

  > For more information about the activation of central logging, see Central Logging of Functionality for Alfabet Components above.

The execution of the monitoring event must be triggered via one of the following options:

- [Triggering Monitoring Event Execution via the Alfabet User Interface](#)

- [Triggering Monitoring Event Execution via a Command Line](#)

Triggering Monitoring Event Execution via the Alfabet User Interface

To trigger an event via the Alfabet user interface, you must have access to the **Event Administration** functionality.

To trigger monitoring event execution in the **Event Administration** view:

1) In the toolbar, click one of the following:

   - **Monitoring Events** > **Trigger Single Monitoring Event** to trigger one-time execution of the monitoring event.

   - **Monitoring Events** > **Trigger Cyclic Monitoring Events** to trigger recurring execution of the monitoring event every ten minutes.

To stop the execution of recurring monitoring events, click **Monitoring Events** > **Stop All Cyclic Monitoring Events**.

Triggered monitoring events are listed in the table of the **Event Administration** functionality in the sections *SYSTEM* > *MonitoringEventSingle* and *SYSTEM* > *MonitoringEventCyclic*.

If the **Event Status** is **Error**, click the Show **Event Message** button to download the error message and correct the issue.

For single monitoring events, the **Event Status** is **Finished** after successful execution. For cyclic monitoring events, the **Event Status** remains **Pending** after successful execution of the monitoring functions, because the event will be immediately re-scheduled for the next execution. The **Start Time** column is updated with the next scheduled start time after each execution.

Triggering Monitoring Event Execution via a Command Line

The execution of monitoring events can be triggered and stopped via the command line tool `AlfaAdministratorConsole.exe`.

Executable	`AlfaAdministratorConsole.exe` located in the ***Programs*** subdirectory of the Alfabet installation directory
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see [Standard Logging for Alfabet Batch Utilities](#).
Command line help	Start executable with `-h` or `-help`

To trigger a one-time execution of a monitoring event, run `AlfaAdministratorConsole.exe` with the following command line:

```
AlfaAdministratorConsole.exe-trigger_monitoring_event-msalias <alias name>-
DbUser <database user name> [-DbPassword <database user password>-timeout
<seconds>]
```

To start the recurring execution of monitoring events, run `AlfaAdministratorConsole.exe` with the following command line:

```
AlfaAdministratorConsole.exe-trigger_monitoring_event_chain-msalias <alias
name>-DbUser <database user name> [-DbPassword <database user password>]
```

To stop the recurring execution of monitoring events, run `AlfaAdministratorConsole.exe` with the following command line:

```
AlfaAdministratorConsole.exe-stop_monitoring_event_chain-msalias <alias
name>-DbUser <database user name> [-DbPassword <database user password>]
```

The table below displays the command line options:

Command Line Option	Mandatory/ Default	Explanation
-<action type>	Mandatory	Set the action type to one of the following: • `-trigger_monitoring_event` for the one-time execution of a monitoring event • `-trigger_monitoring_event_chain` for the recurring execution of monitoring events • `-stop_monitoring_event_chain` to stop the recurring execution of monitoring events.
-msalias <alias name>	Mandatory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.
-msalias- esfile <Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
-DbUser <database user name>	Mandatory	Specify the user name for access to the Alfabet database on the database server. For Oracle® databases, the user name is identical to the schema name.
-DbPassword <database user password>	Mandatory	Specify the password for access to the Alfabet database on the database server.

Carrying Out Database Maintenance Tasks

The Alfabet database is the basis of the Alfabet application. Software AG provides mechanisms to migrate configurations done on one database to other databases and update databases to new releases and patch releases of Alfabet. This chapter describes all database-related tasks that must be performed on demand or on a regular basis on the Alfabet database:

- Backing Up and Recovering the Alfabet database

- Rebuilding Defragmented Indices

    - Index Defragmentation on Server Start

    - Index Defragmentation via a Command Line Job

    - Index Defragmentation During ADIF Import and Event Execution

- Administration Tasks Related to Solution Design

    - About the Storage of the Alfabet Meta-Model in the Alfabet database

    - Activating Update of ALFA_MM_*_INFO and ALFA_PM_*_INFO Tables During Update of the Meta-Model via AMM File

    - Storing the Configuration of a Database in an AMM Update File

    - Restoring the Configuration of the Alfabet Solution Environment

        - Restoring the Configuration of the Alfabet Solution Environment with the Alfabet Administrator

        - Restoring the Configuration of the Alfabet Solution Environment via Console Application

- Managing Assemblies

    - Uploading and Managing Assemblies with the Alfabet Administrator

        - Executing a Script

    - Uploading Assemblies via the *.amm Update File to Another Database

Backing Up and Recovering the Alfabet database

It is highly recommended that you back up the Alfabet database at least once a day. Typically, a hot backup is done on the level of the database server.

Backup on the file system is typically not required as the `AlfabetMs.xml` configuration file is the only element of the standard configuration that is kept on the level of the file system. Depending on the specific customer environment, other configuration files such as job schedules may also be stored directly on the level of the file system.

Rebuilding Defragmented Indices

Adding data to and deleting data from database tables leads to database index fragmentation which has a negative impact on performance. Especially ADIF imports and rescan of indicators via the command line tool `RescanIndicatorsConsole.exe` or via the **Job Scheduler** functionality may trigger a high number of single database insert and delete transactions. As a result, CPU usage is increased, and performance is reduced. It is recommended to rebuilt indices in regular intervals for object classes subject to ADIF import or rescan of indicators.

The following mechanisms are available for index defragmentation:

- Index Defragmentation on Server Start

- Index Defragmentation via a Command Line Job

- Index Defragmentation During ADIF Import and Event Execution

Index Defragmentation on Server Start

Index defragmentation can be executed automatically each time the Alfabet Server is started or when the meta-model is updated via AMM files. Defragmentation on server restart requires activation in the server alias:

1) In the explorer of the Alfabet Administrator, click the **Alfabet Aliases** node.

2) In the table, click the server alias that you want to configure.

3) In the toolbar, click the **Edit** 🖉 button. An editor opens.

4) Go to the **Database Settings** > **Details** tab.

5) Check the **Rebuild fragmented indices on server start** attribute.

Index Defragmentation via a Command Line Job

Index defragmentation can be triggered via the command line tool `AlfaAdministratorConsole.exe.`

To rebuild indices for a number of object classes defined by object class name, use the following command line for the `AlfaAdministratorConsole.exe`:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <user password>-rebuild_indices-class_names
<comma separated list of object class names>
```

To rebuild indices for a number of database tables identified by table name, use the following command line for the `AlfaAdministratorConsole.exe`:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <user password>-rebuild_indices-table_names
<comma separated list of database table names>
```

To rebuild indices for all database tables, use the following command line for the `AlfaAdministrator-Console.exe`

The table below displays the command line options:

Command Line Option	Mandatory/ Default	Explanation
`-rebuild_class-indices`	Mandatory	To rebuild indices for database tables in the Alfabet database, start the console application with `-rebuild_indices`
`-class_names` <comma separated list of object class names>	It is mandatory to specify either-`class_names`,`-table_names`, or-`all`	Define the object classes for that an index shall be rebuild as a comma separated list of object class names. For example: `-class_names ApplicationGroup,OrgaUnit`
`-table_names` <comma separated list of database table names	It is mandatory to specify either-`class_names`,`-table_names`, or-`all`	Define the database tables for that an index shall be rebuild as a comma separated list of database table names. For example: `-table_names APPLICATIONGROUP, RELATIONS`
`-all` <TRUE/FALSE>	It is mandatory to specify either-`class_names`,`-table_names`, or-`all`	Add this command line parameter and set it to `TRUE` to rebuild all indices.
`-msalias` <alias name>	Mandatory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.
`-msaliasesfile` <Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName` <Alfabet user name>	Mandatory	Alfabet user name of a named Alfabet user for login. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPassword` <user password>	Optional	Alfabet login password.

Index Defragmentation During ADIF Import and Event Execution

Execution of ADIF import can lead to a high database table index defragmentation level when the batch import of data to a database table is performed.

ADIF import schemes have been amended with an analysis of which database tables are read or changed when the ADIF scheme is executed. the analysis not only provides an overview of the database tables for that index fragmentation may occur during execution of the ADIF import. It also provide a means to configure the ADIF scheme to trigger index defragmentation for specified database tables as part of the import mechanism:

The analysis is added in the **Table Usage** sub-node of the ADIF scheme. The **Rebuild Indices on Completion** attribute is available for each database table that the ADIF scheme will write data to. If the attribute is set to `True` for a database table, the index of this database table will be re-built automatically as part of the ADIF import.

1) In the ADIF Explorer in Alfabet Expand, expand the node of the ADIF import scheme.

2) Expand the Table Usage node.

3) For each object class that you would like to trigger index defragmentation after import, click the object class node in the **Table Usage** node and set the **Rebuild Indices On Completion** attribute to `True`.

> The **Rebuild Indices On Completion** attribute is only available in nodes ending with **(Write)** which indicates that the ADIF job will change data in the database table of the object class.

Event tables triggering ADIF import will inherit the **Table Usage** analysis from the defined ADIF import scheme. The **Rebuild Indices on Completion** attribute is editable in the event template to adapt the setting to the current use case.

Administration Tasks Related to Solution Design

For security reasons, it is recommended that all solution configuration changes are performed in a development environment, and tested in a test environment before migrating them to the production environment. All development and test environments should be based on copies of the production database. An overview of the environments is given in section Best Practice Installation and Workflow.

For best practice solution configuration and database maintenance, it is recommended that a solution designer should be responsible for the development database, whereas any tasks performed on the database in other environments should be performed by a system administrator. The tool Alfabet Administrator allows all tasks that are required for database maintenance to be performed by the system administrator. Most of the tasks can also be performed by the solution designer using Alfabet Expand. Depending on your enterprise's policies, the solution designer could optionally carry out system administration tasks on the development database independent of the system administrator.

Software AG provides a mechanism that allows the customer configuration to be saved and restored in another database independent of other parts of the database:

The customer configuration store and optionally the solution store can be saved to an AMM updater file. The updater file allows the configuration of a target database to either be merged or replaced by the configuration stored in the AMM updater file.

> (i) AMM is a proprietary file format and therefore not recognized by web browsers during download. If you want to store AMM files in a web based content management system or intranet for download, you must create a zip file containing the AMM file and store the ZIP file so that you can download the file via a Web browser to your local file system.

You can save all or part of the customized meta-model configuration for the Alfabet solution to AMM update files. An AMM file can then be used to replace or modify the configuration in an existing database. Not only is information about the database configuration contained in the AMM files. The AMM files also control the way the information is updated in the target database. The decision to replace or merge the configuration with that of the target database or to migrate workflows during the configuration update is made when the AMM file is created.

`*.amm` files are typically created with the tool Alfabet Expand. The solution designer that performed the configuration is best aware of the required changes to apply the configuration to the target database. For more information about the creation of the update file in Alfabet Expand and about special mechanisms available to the solution designer to selectively add configuration objects to the AMM file, see the reference manual *Configuring Alfabet with Alfabet Expand*.

System Administrators typically restore the configuration provided by a solution designer in an AMM file to provide the configuration changes in the production environment. Nevertheless, a subset of the functionalities to create an AMM file is also available to system administrators in the Alfabet Administrator. The mechanism can be used to copy configurations to a test environment or to deliver a configuration that need to be reviewed to Software AG Support.

A recommended procedure for the development of new configurations is described in the figure below:
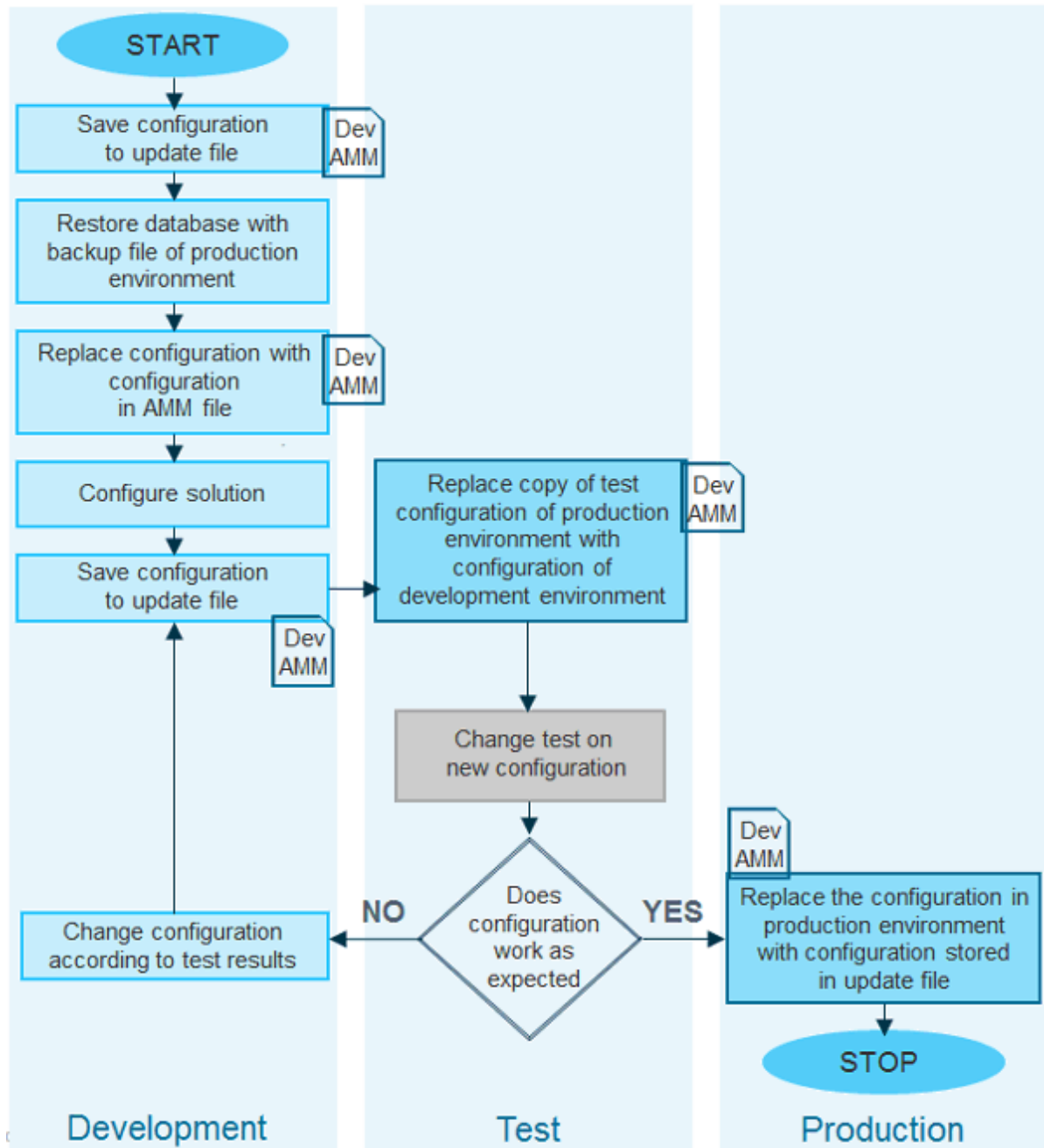
FIGURE: Example for a best practice workflow for the configuration of the Alfabet solution

The process involves the following:

- **Copying the production database to the development database**:

  A new configuration is best performed on an up-to-date copy of the production database. Backup files created during regular backup on database level can be used to restore a copy of the production database in a test environment.

  The following options are available:

  - The solution designer can restore the current database and start working with the exact copy of the production database, thus overwriting any configuration steps done in the development environment that have not yet been applied to the production environment

- The solution designer can conserve current changes to the configuration in the development environment. He/she can save the current customized configuration of his/her development database to an AMM file before overwriting the development database with the database backup file of the production database and restore the customized configuration he/she is currently working on from the AMM after database restore.

- **Saving the configuration and restoring it first in a test environment and later to the production environment**:

  After performing the required solution configurations, the solution designer saves the configuration to an AMM update file. This file is then used to replace the configuration of an up-to-date copy of the production database in a test environment. Only after tests have ensured that the configuration is correct, the configuration of the production database can be replaced or merged with the new configuration.

 Whether configurations must be merged or replaced with a new configuration depends on the configuration steps performed. The AMM file includes information about the update mechanism. It is recommended that the solution designer directly generates the AMM file in Alfabet Expand because the solution designer can best decide on the update requirements to be specified for the AMM file.

About the Storage of the Alfabet Meta-Model in the Alfabet database

The database comprises the following information:

- The functionality of the Alfabet solution (solution store). The software functionality is delivered by Software AG as a DLL file. The main solution assembly is typically uploaded to the Alfabet database by Software AG Support only during the upgrade to new Alfabet versions. Special assemblies delivered to the customer upon request can be uploaded using the **Assemblies** functionality in the tool Alfabet Administrator. For more information about managing assemblies, see the section Managing Assemblies in the reference manual *System Administration*. Assemblies uploaded to one Alfabet database can then be stored to AMM files and transferred to other Alfabet database s.

- The configuration and database structure provided by Software AG for the Alfabet solution (configuration store). This meta-model information is typically updated by Software AG Support only during upgrade to a new Alfabet version. The update is executed via an AMM file that includes all meta-model related information.

- The configuration performed by the customer using Alfabet Expand or Alfabet Administrator (customer configuration store). Customers can save the configuration in AMM files and merge the configuration to another Alfabet database or overwrite the existing configuration with the customer configuration stored in an AMM file. For more information about saving and restoring the configuration store, see Restoring the Configuration of the Alfabet Solution Environment.

- The instances in the Alfabet inventory created by users in the Alfabet user interface (instance store). The instance store cannot be stored in AMM files. Nevertheless, the instance can include configuration relevant data such as, for example, reference data that includes indicator types and role types defined via the **Configuration** functionalities in the Alfabet user interface. To transfer data over to another Alfabet database, a direct connection between the source and target database can be configured by the solution designer and the data can then be integrated into the target database via the Alfabet user interface of the target database. Integration can be performed on an object-by-object basis.

- The documents uploaded to the **Internal Document Selector** (document store). The customer-defined guide pages that are created to provide guide pages for the Alfabet user interface are the only part of the document store that can be saved to an AMM file that can then be used to overwrite guide pages of a target database with the version stored in the AMM file. All other documents in the **Internal Document Selector** can only be stored in normal database backup files on the database server level.
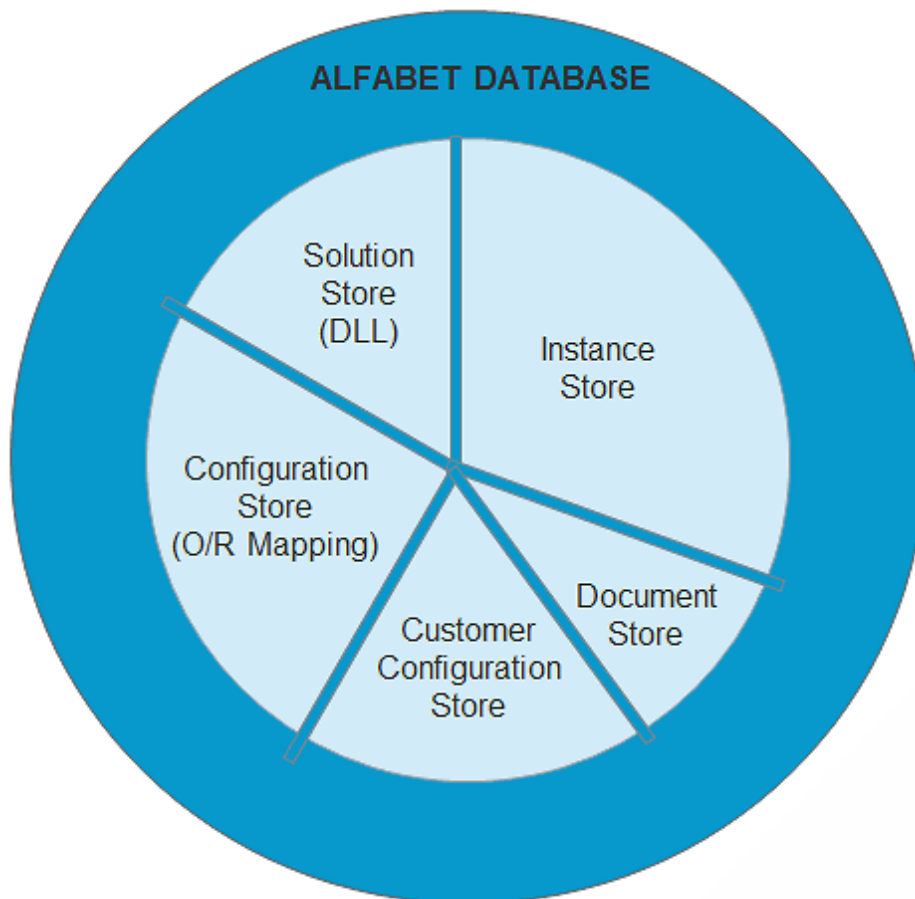
Activating Update of ALFA_MM_*_INFO and ALFA_PM_*_INFO Tables During Update of the Meta-Model via AMM File

Several object classes in the Alfabet meta-model store information about the current structure of the object class model including both standard and customer defined configurations performed on Alfabet object classes as well as the current configuration of the presentation model, like the configuration of object profiles. The information stored in these tables can be used to build configured reports that inform about selected aspects of the configuration.

The names of the object classes storing meta-model configuration information all start with `ALFA_MM_` and end with `_INFO`, like for example `ALFA_MM_CLASS_INFO`. The names of the object classes storing presetnation model information all start with `ALFA_PM_` and end with `_INFO`, like for example `ALFA_PM_OB-JECTVIEW_INFO`

The database tables of these object classes are not updated automatically after change of the configuration done via the configuration tool Alfabet Expand. Solution designers must update the information via button interaction after configuration changes are done.

> ⓘ For more information about the `ALFA_PM_*_INFO` object classes and the update of the information via button interaction, see *Getting An Overview of the Current Configuration* in the reference manual *Configuring Alfabet with Alfabet Expand*.

Update of the meta-model via AMM file will alter the configuration and the `ALFA_MM_*_INFO` and `ALFA_PM_*_INFO` tables should be updated after an update of the meta-model via AMM file. This update can either be done via the button interaction in Alfabet Expand or the Alfabet Web Application can be configured to update the tables automatically after the configuration has been updated via AMM file. By default, `ALFA_MM_*_INFO` tables will be updated automatically while `ALFA_PM_*_INFO` tables are not updated.

Automatic update of the `ALFA_MM_*_INFO` and `ALFA_PM_*_INFO` tables is activated in the server alias configuration of the Alfabet Web Application using the tool Alfabet Administrator:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer.

2) In the table, select the server alias that you want to edit and click the **Edit** 🖉 button in the toolbar. An editor opens.

3) In the **Expand** tab, select the checkmark of the following options to activate the respective functionality:

- **Automatically generate ALFA_MM tables**

- **Automatically generate ALFA_PM tables**

4) Click **OK** to save your changes.

Storing the Configuration of a Database in an AMM Update File

The storage of configuration data in an AMM file is typically done by the solution designer that performed the configuration to make sure that all relevant parts are included. Alfabet Expand offers all relevant mechanisms for creation of AMM files.

Optionally AMM files can also be created by a system administrator using the tool Alfabet Administrator:

1) In the **Alfabet Aliases** section of the **Administrator** explorer, right-click the server alias of the Alfabet Server connecting to the database containing the content to be modified and select **Connect**. A login window is displayed.

2) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

3) Right-click the server alias again and select **Create Configuration Meta-Model Update File**. The **Create Configuration Meta-Model Update** editor opens.

4) In the **Meta-Model Content** tab, select the configuration that you want to add to the AMM file:

- **Save Configuration**: Select the checkbox to save custom object class properties, custom enumerations and all customer defined configuration objects in the **Presentation**, **Functions**, and **Surveys** tabs in Alfabet Expand.

- **Save Reports**: Select the checkbox to save the content of the **Reports** root node of the explorer in the **Reports** tab in Alfabet Expand. If the report structure in the target database will change with the update, bookmarks linking to configured reports will be automatically updated when the configuration is merged.

  > (i) If a configured report that exists both in the target database and in the AMM file is updated, and a user or solution designer has restricted the display of columns of the configured report via the **Configure** button, new columns added to the configured reports will be hidden because they are not selected in the visibility setting.
  >
  > For information about hiding columns in configured reports, see the section *Defining the Visibility of Page Views/Configured Reports Available at the Root Node of an Explorer* in the chapter *Configuring User Profiles for the User Community*.

- **Save Workflow Templates**: Select the checkbox to save the content of the explorer of the **Workflow** tab of Alfabet Expand.

- **Save ADIF Schemes**: Select the checkbox to save the content of the explorer of the `ADIF` tab in Alfabet Expand.

  > (i) After the target database is updated by means of the AMM file, all ADIF import schemes in the target database that have the **Auto-Run** attribute set to `True` will be automatically executed. For more information about automatic execution of ADIF schemes, see *Configuring ADIF Schemes to be Automatically Executed on Update of the Meta-Model* in the reference manual *Alfabet Data Integration Framework*.

- **Save Event Templates**: Select the checkbox to save the content of the explorer of the **Events** tab in Alfabet Expand.

- **Save Diagrams**: Select the checkbox to save the custom diagram definitions and custom diagram item templates defined in the **Diagrams** tab in Alfabet Expand.

  > ⚠ Icons are not automatically saved if they are included in a diagram item template and the diagram item template is stored in the AMM file. If your configuration includes icons that are not available in the target database, make sure to also select **Save Icons**.

- **Save Publications**: Select the checkbox to save the content of the **Publications** tab in Alfabet Expand.

- **Save Database Views**: Select the checkbox to save the database views defined in the **Meta-Model** tab in Alfabet Expand.

  > (i) When the configuration is restored to the target database, all database views in the target database that are created based on a **Database View** configuration object will be recreated, even if **Save Database Views** is not checked. The solution designer should then check whether the creation of database views was successful. The required procedure is described in the section *Saving the Configuration of the Alfabet Solution to an AMM File* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- **Save Icons**: Select the checkbox to save the icons uploaded to the Alfabet database.

> (i) Please note that the **Save Icons** checkbox in the **Meta-Model Content** tab must be selected in order to include any images included in the configuration of guide views.

- **Save Culture Settings**: Select the checkbox to save the content of the **Cultures** and **API Cultures** explorer nodes of the **Meta-Model** tab in Alfabet Expand.

- **Save Translation**: Select the checkbox to save the custom translation of strings displayed in the Alfabet user interface that are customized via the **Translation Editor** in Alfabet Expand.

  > (i) When the translation includes the caption and description of configured reports, the **Update Translation** functionality available in the **Report** root node in Alfabet Expand must be executed on the target database after update with the AMM file.

5) Select the following options, if applicable for your configuration:

- **Replace Entire Existing Configuration**: Select the checkbox if you want the configuration in the AMM update file to overwrite an existing configuration in the target database during the **Update Meta-Model** action. If the checkbox is not selected, the configuration in the AMM update file is merged with the configuration in the target database.

  > ⚠ If the **Replace Entire Existing Configuration** checkbox is selected, the configuration of the target database will be deleted prior to saving the configuration in the AMM file to the target database. All customer configurations that are part of the configuration of the target database but not part of the configuration saved in the AMM file will be lost. The replace mechanism deletes all parts of the configuration including the parts that are not selected to be part of the AMM update. The **Replace Entire Existing Configuration** checkbox should only be selected for AMM files when the parts of the configuration to be replaced have been selected.
  >
  > Please note that reference data, assemblies and guide pages are not deleted prior to applying the configuration stored in the AMM file.
  >
  > Cultures in the target database are removed with the exception of the primary culture (en-US), the default culture and the configuration culture. Any settings in the AMM file about the default culture or the configuration culture are ignored and the settings in the target database are maintained.
  >
  > Configured reports that are automatically generated for the faceted semantic search functionality of the AlfaBot will not be deleted from the target database. For more information about automatically generated reports, see *Managing Automatically Generated Reports*.
  >
  > If you want to replace only part of the configuration (for example, the configuration of workflows only), it is recommended that you use the solution tagging option. For more information about solution tagging, see *Identifying Configuration Objects via Solution Tagging* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- **Automatically Migrate Affected Workflows**: This checkbox is only selectable if you have included workflow template configuration in the AMM file.Select the checkbox if you want the workflow templates to be automatically migrated to the target database. Migration of workflows is a complex process that is described in detail in the section *Creating a Migration Definition to Update Running Workflows* in the chapter *Configuring Workflows*.

- **Only Include Objects with the Following Tags**: The box lists all solution tags available in the current configuration. If you do not select any checkboxes, all configuration parts selected in the **Meta-Model** tab are is saved without taking solution tagging into account. If you select one or multiple checkboxes, only solution objects which are both marked with a selected tag

and are of a configuration object type selected in the Meta-Model tab are included into the AMM file. If solution tagging is not available for a configuration object type that is selected, all objects of that type will be saved independent of selected tags. For more information about solution tagging, see the section *Identifying Configuration Objects via Solution Tagging* in the reference manual *Configuring Alfabet with Alfabet Expand*.

> For example, the configuration includes 20 configured reports and 5 publications.
>
> You have tagged two configured reports and two publications with a tag **Publica-tionReports** and select this tag in the **Only Include Objects with the Following Tags** field.
>
> In addition, you have selected the **Save Reports** checkbox, but you have not se-lected the for **Save Publications** checkbox.
>
> The AMM file will include the two tagged configured reports. It will not include any other configured reports because they are not tagged. It will not include an tagged or untagged publications because publications were not selected for upload.

- **Provide comma-separated tags to find objects to be removed**: Enter the name of a solution tag or a comma-separated list of multiple solution tag names to delete all public (customer-defined) configuration objects tagged with the specified tag name(s) from the target database prior. This option is useful if a tagged configuration has been changed in a way that includes the deletion of objects. The obsolete objects can be removed form the target database as well. For more information about solution tagging, see the section *Identifying Configuration Objects via Solution Tagging* in the reference manual *Configuring Alfabet with Alfabet Expand*.

> Make sure that all objects required for the tagged solution are available in the AMM file when setting this option. All objects with the solution tag not included in the AMM file will be deleted from the target database.

> If the tag name(s) that shall be used to remove objects is also used in the database you are currently connected with to tag configuration objects, you can select the tag name(s) from a multi-select combo-box instead of typing them in the field. Click the button on the right of the field to open the multi-select combo-box and select the checkbox of all tags in the current configuration for that all configuration ob-jects shall be deleted in the target database prior to import of the configuration ob-jects in the AMM file.

6) The **Guide Pages** tab displays all guide page projects in the current database. Select the checkbox for all guide page projects that should be merged to the existing guide page configuration of the target database. Guide pages are merged to the target database independent of the setting of the **Replace Entire Existing Configuration** field in the **Meta-Model** tab. Optionally, you can select the **Remove All Guide Pages from Target Database Before Updating** checkbox to overwrite the complete guide page configuration of the target database with the guide pages in the AMM file. If the checkbox is not selected, guide pages in the AMM file will be added to existing guide pages in the target database, thus overwriting guide pages with the same name.

> Please note that the **Save Icons** checkbox in the **Meta-Model Content** tab must be se-lected in order to include any images included in the configuration of guide views.

⚠ The styles configured in the guide pages stored in the AMM file will overwrite the styles of the guide pages in the target database. You should ensure that the styles relevant for your enterprise are correctly configured in the guide pages before importing them via an AMM file to the production environment. For more information about the configuration of guide pages and their styles, see the section *Formatting and Designing the Guide Pages* in the reference manual *Designing Guide Pages for Alfabet*.

7) In the **Reference Data** tab, user profiles and a specified subset of configurations performed in the **Configuration** functionalities in the Alfabet user interface can be optionally uploaded to the AMM file. These objects are stored in the instance store of the database. In contrast to the other configuration objects, they can be created and edited by users via the Alfabet user interface. Independent of the setting of the **Replace Entire Existing Configuration** field in the **Meta-Model** tab, they are always merged with the existing configuration in the target database.

ⓘ Note the following about the update of user profiles and reference data via AMM files:

- **User profile configuration**: User profiles in the AMM file are added to the existing user profile configuration in the target database. User profiles with the same name will be overwritten in the target database.

  ⚠ If a guide page/guide view is defined as the start page for a user profile, you must explicitly include the guide page/guide view in the AMM update. If the guide page/guide view is not included, then the user will see an empty start page.

- **Configuration of reference data and evaluations**: Existing objects in the Alfabet database are overwritten if the key property for identification of the object is identical. The following table lists the configured objects that can be migrated via AMM, the key used to identify the objects, and whether the objects can be added to the AMM file by direct selection of objects or as sub-objects of a selectable parent object:

Object Class	Key	Selection
EvaluationType	Name	Directly
IndicatorType	Evaluation Type, Name	Included with selection of the evaluation type they are assigned to.
RoleType	Name	Directly
Prioritiza-tionScheme	Name	Directly
ITPortfolio	Name	Directly

DiagramView	Name	Directly
DiagramView-Item	Name	Included with selection of the diagram view they are assigned to.
ColorRuleGroup	Name	Directly
ColorRule	Name	Directly or included with selection of the color rule group they are assigned to.
ClassConfiguration	Class Name	Directly

A table lists all configuration objects available for the type of configuration object selected in the **Select Class** field above the table. Select the type of configuration object in the **Select Class** field and click the cell in the **Save** column of the table for all configuration objects of the selected type that are to be saved to the AMM file:
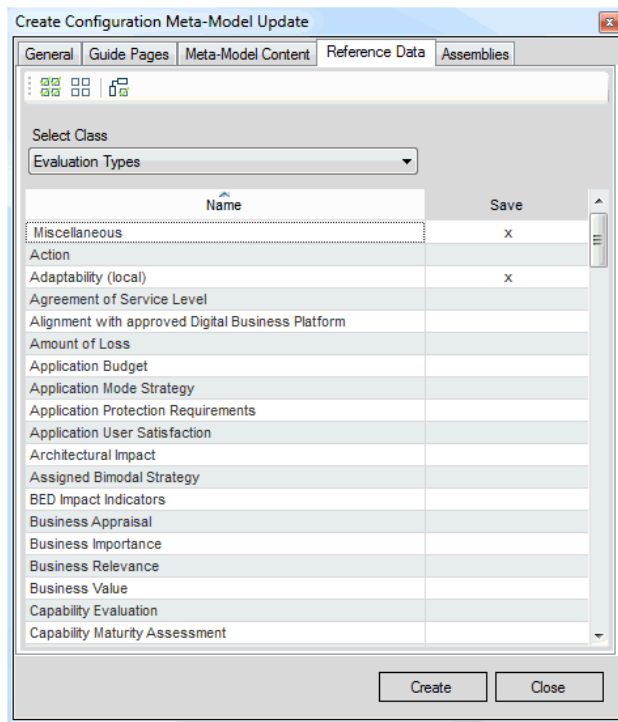
To save all configuration objects or user profiles currently displayed in the table to the AMM file, click the **Check All** ⊞ button. Click the **Uncheck All** ⊟ button to clear the selection of all objects in the table.

When you select **Class Configuration** in the **Select Class** field and stereotypes are defined for an object class, each stereotype is listed in a separate row of the table with the **Name** defined as "Name (Stereotype)".

Color rules that are assigned to a color rule group cannot be included separately. Select **Color Rule Groups** in the **Select Class** field and select a color rule group to include the color rule group and all color rules that are assigned to the color rule group to the AMM update file. When you select Color Rules in the Select Class field, the table only displays color rules that are not assigned to a color rule group. These color rules can be selected separately.

Select the **Check Dependent Objects** ⬚ button to select objects that depend on other objects that you have currently selected in the list. When you click the button, the dependent objects will be automatically selected:

- For each evaluation type currently selected in the list, the following dependent objects are also selected:

  - all prioritization schemes to which the evaluation type or any of the indicator types assigned to the evaluation type are assigned

  - all IT portfolios to which the evaluation type or any of the indicator types assigned to the evaluation type are assigned

  - all diagram views to which any of the indicator types assigned to the evaluation type are assigned

  - class configurations to which the evaluation type is assigned

- For each prioritization scheme currently selected in the list, the following dependent objects are also selected:

  - all IT portfolios to which the prioritization scheme is assigned

  - class configurations to which the prioritization scheme is assigned

- For each IT Portfolio currently selected in the list, the following dependent objects are also selected:

  - class configurations to which the IT portfolio is assigned.

8) In the **Assemblies** tab, assemblies can be optionally uploaded to the AMM file if the configuration has been specified via a DLL file delivered by Software AG. If you need to include assemblies in the AMM file, see the section *Managing Assemblies* for detailed information.

9) Click the **Create** button to generate the AMM update file. A message will be displayed once the AMM has been successfully created.

Restoring the Configuration of the Alfabet Solution Environment

The configuration in AMM update files can be restored in target databases using the Alfabet Administrator, a console application or Alfabet Expand.

Please note the following about the process to update the meta-model with AMM files:

- During update of the meta-model with an AMM file the database is run in a restricted mode. That means that all connections to the database except for the one required to update the meta-model are closed and new connections cannot be established. The restricted mode ends automatically when the update process is finished. If the restricted mode persists after the update is finished, it can be manually released via the Alfabet Administrator. For more information about manually releasing the restricted mode, see Releasing the Restricted Mode in the reference manual *System Administration*.

- Shut down all other Alfabet components except the database server hosting the Alfabet database prior to update of the meta-model. For more information about planned shutdown of the Alfabet components, see Planned Outages of the Alfabet Components in the reference manual *System Administration*.

- Prior to update of the meta-model, database replication mechanisms targeting the Alfabet database must be shut down.

- Login to the Alfabet database on the database server for database update is not performed with the database user defined in the server alias configuration, but rather by means of the database user `AlfaRuntimeUser` that is automatically generated for the process by the system. This database user has Read/Write access to the database during the update of the meta-model. It is not possible to access the Alfabet database with a user `AlfaRuntimeUser` in other contexts.

- A number of mechanisms are available that ensure that the translation settings of the target database are not corrupted via a meta-model update with an AMM file:

  - Information about the configuration language of the source database is stored in the AMM file. Update of the meta-model via the AMM file will fail if the settings for the configuration language are different in the AMM file and the target database.

  - On update of the meta-model with an AMM file which is configured to replace the culture configuration of the target database, the cultures in the target database are removed with the exception of the primary culture (en-US), the default culture and the configuration culture. Any settings in the AMM file about the default culture or the configuration culture are ignored and the settings in the target database are maintained.

- The case sensitivity setting from the collation of the source database is stored in the AMM file. Update of the meta-model via the AMM file will fail if the case sensitivity setting in the AMM file differs from the case sensitivity setting of the target database.

- Length limitation for database table columns on an Oracle® database server is more restrictive than on a Microsoft® SQL Server®. The allowed length on Oracle® database servers for the **Tech Name** attribute of custom object class properties in Alfabet depends on the configuration of data translation. The maximum length of a technical name may not exceed 25 characters for properties that have the **Enable Data Translation** attribute set to `True`. For properties that are not translatable, the maximum length of a technical name may not exceed 30 characters. To avoid problems on migration of the database from a Microsoft® SQL Server® to an Oracle® database server because of **Tech Name** attributes of custom object class properties not matching the restrictions, a check for database server compatibility is available. A read-only **Database Platform Compatibility** attribute is available in the **Environment** node in the **Admin** tab. The attribute is set to `Oracle` if the database is hosted on an Oracle® database server. If the database is hosted on a Microsoft® SQL Server®, the **Database Platform Compatibility** attribute is set to `SqlServer` if the database contains **Tech Name** attributes that violate the size restriction and will be set to `Common` if the database does not violate the size restriction. On creation of an AMM file, the **Database Platform Compatibility** attribute of the current database is written to the AMM file. On update of the meta-model with the AMM file, the **Database Platform Compatibility** attribute

value in the target database is compared with the value in the AMM file. Updating the meta-model will fail if the Alfabet database is hosted on an Oracle® database server and the AMM file has the **Database Platform Compatibility** value set to `SqlServer`. The database will remain unchanged and offending **Tech Name** values will be written to a log file.

⚠️ **Before you update a production database with an AMM file created with an Update Solution option, you must backup the target database for security reasons.** Solution tagging is a sensitive process that can lead to data loss if not carefully planned and executed. It is strongly recommended that you perform solution tagging with the help of your consultant from Alfabet only. The update of a database via an AMM file created with an **Update Solution** option should be thoroughly tested on a test database before the production environment is updated. For more information about solution tagging, see the section *Identifying Configuration Objects via Solution Tagging*.

If you merge the solution configuration in the AMM file with the configuration of a target database, only the configuration objects that are stored in the AMM file will be overwritten. Overwriting configuration objects is especially important for the update of configuration objects with sub-objects. For example:

- Updating an object class with information in the AMM file will delete all custom object class properties and numeration information for the class in the target database and substitute it with the information in the AMM file.

- Updating a report folder will delete all reports from the report folder in the target database and substitute it with the reports available in the AMM file configuration.

Therefore, prior to creating an AMM update file, you must ensure that the configuration in the development database is identical to the configuration in the target database (with the exception of the desired configuration change).

Update of the meta-model with an AMM update file changes the configuration of the meta-model in your database. Please note the following about the restore of the configuration:

- **Clarify any concerns regarding your specific customization before initiating the Update Meta-Model functionality!** Some customizable features in Alfabet may not be part of the meta-model configuration. They are stored in the instance store and could be lost/damaged when a configuration is saved and restored to another database with an AMM file. This applies to the following configurations:

  - **Export definitions**: To save and restore the export definitions, the export definitions must be saved and restored separately using the tool Alfabet Expand. In Alfabet Expand, the context menu of the root explorer nodes for the respective configurations offer a **Save as** functionality that allows you to save the specific part of the configuration as an XML file and a **Read from File** functionality and **Merge from File(s)** functionality to restore the specific configuration.

  - **Mandate configuration**: The configuration of mandates is saved in the instance store of the database and can only be saved and read/merge to another database by using the **Import Data Search** functionality or by manually recreating the configuration in the target database using Alfabet Expand or the Alfabet Administrator. For more information about the activation and use of the **Import Data Search** functionality, see the section *Importing Objects of Configuration Relevant Object Classes from a Master Database* in the reference manual *Configuring Alfabet with Alfabet Expand*.

  - **Most reference and evaluation data configured in the Configuration functionalities in the Alfabet user interface**: For example, reference data such as role types, cost types , indicator types, or portfolios are configured in Alfabet configuration functionalities in the

Alfabet user interface. These configurations are saved in the instance store of the database. Only the following subset of these configurations can be included in the AMM file:

- Evaluation Types

- Portfolios

- Prioritization Schemes

- Diagram Views

- Color Rule Groups

- Color Rules

- Class Configurations

- Roles

Other configuration data can only be saved and read/merge to another database by using the **Import Data Search** functionality or by manually recreating the configuration in the target database using Alfabet Expand or the Alfabet Administrator. For more information about the activation and use of the **Import Data Search** functionality, see the section *Importing Objects of Configuration Relevant Object Classes from a Master Database* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- **Documents uploaded to the Internal Document Selector**: Documents in the **Internal Document Selector** can only be stored in normal database backup files on the database server level or added to the target database manually. This may impact the images and style sheet files uploaded to be used for visualization items (configured HTML templates for the workflows functionality). While the visualization item can be saved to an AMM file and uploaded to a target database via meta-model update, the images and style sheet files must be uploaded to the **Internal Document Selector** of the target database via Alfabet Expand or the Alfabet user interface.

- All database views in the target database are recreated after a meta-model update and need to be checked for consistency afterwards as described in the section *Checking the Consistency of the Database View With the Meta-Model*. The recreation ensures that new database views added to the configuration via the meta-model update are created and that database views that are not compatible with the changes to the meta-model applied via meta-model update are removed.

- AMM files can be configured to either replace or merge the configuration stored in the AMM file with the configuration in the target database:

- **Replace Configuration**: The whole customer configuration in the target database will be deleted prior to writing the content of the AMM file to the target database. Configurations that have no corresponding object in the AMM file will be deleted.

  > (i) Configured reports that are automatically generated for the faceted semantic search functionality of the AlfaBot will not be deleted from the target database. For more information about automatically generated reports, see *Managing Automatically Generated Reports*.

- **Merge Configuration**: All objects in the configuration will overwrite corresponding objects in the database. Database objects that have no corresponding object in the AMM will remain unchanged. Objects that are only available in the AMM will be added.

Alfabet provides mechanism to check the content of an AMM file prior to update of the meta-model and to control the success of the update action:

- When updating the meta-model using Alfabet Expand or the Alfabet Administrator, a window for selection of the AMM file that shall be used for update opens. After having selected an AMM file, you can click the button **View Content Summary** to see a summary of the content of the AMM file.

- You can use the **Compare Configurations** functionality of Alfabet Expand to view the differences between the configuration in the AMM file with the configuration of the target database that you are planning to replace or merge. For more information, see *Comparing Database Configurations* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- A log file is created during update of the meta-model. The log file is written to the directory that contains the AMM file and has the name <Name of the AMM file>_<Time Stamp>.log. You can consult the log file to view any issues that occurred during update.

- After update, you should check the consistency of the update with the standard meta-model. A consistency check can be performed via Alfabet Expand. For more information, see in the reference manual *Configuring Alfabet with Alfabet Expand*.

Restoring the Configuration of the Alfabet Solution Environment with the Alfabet Administrator

Ensure the following prior to using this functionality:

- Make sure that no Alfabet components are currently connected to the Alfabet database. This applies to the Alfabet Web Application, the Alfabet Server (service), Alfabet Expand or batch utilities, for example. These components should be stopped prior to the update meta-model action and re-started afterwards. For information about shutting down the components, see Planned Outages of the Alfabet Components.

- Clarify all configuration issues that are listed in the section Restoring the Configuration of the Alfabet Solution Environment.

- Backup the target database before starting the update process.

Open the Alfabet Administrator and perform the following tasks:

1) In the **Alfabet Aliases** section of the **Administrator** explorer, right-click the server alias of the Alfabet Server connecting to the database containing the content to be modified and select **Connect**. A login window is displayed.

2) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

3) Right-click the server alias again and select **Update Meta-Model**. A new window opens:

4) Click the **Browse** ![...] button next to the **Enter the path of the file required to update theAlfabetmeta-model** field. An explorer opens.

5) Navigate to the folder where the *.amm file with your saved configuration is located and click **Open**. The path to the update file is displayed.

The following information about the selected file is displayed after the file was selected:

- If a name was specified during the creation of the update file, it will be displayed in the **Operation** field.

- If a description was specified during creation of the update file, it will be displayed in the **Description** field.

- The relevant field of the **Replace Existing Configuration** or **Merge Existing Configuration** options is automatically selected depending on whether a merge or replace action is being executed.

- To view a summary of the content of the file, click **View Content Summary**. A new window opens that provides information about the options selected during creation of the file.

6) Optionally, you can change the location for the log file created during the update process in the **Log File** field. By default, the log file is stored in the directory of the *.amm file used for the update. The default log file is automatically specified in the **Log File** field.

> ℹ️ Changes in the vocabularies that might affect custom translations are logged separately in a Microsoft Excel file. The log file for the update process includes a link to the translation log.

7) Click **Update**. The configuration stored in the update file replaces or merges with the configuration of your current database.

> ℹ️ After update of the configuration, all ADIF import schemes that are available in the target database after the update and have the **Auto Run** attribute set to `True` will be automatically executed.
>
> If the automated data translation feature is implemented, update of the meta-model will take significantly longer than without the feature being implemented.

> After applying a configuration to a database, it is recommended that you check the database for consistency with the meta-model as described in the section *Saving the Configuration of the Alfabet Solution to an AMM File* in the reference manual *Configuring Alfabet with Alfabet Expand*.

Restoring the Configuration of the Alfabet Solution Environment via Console Application

The update of a configuration via a configuration stored in a *.amm file can be performed via a command line tool provided by Software AG:

Executable	`AlfaAdministratorConsole.exe` located in the ***Programs*** subdirectory of the Alfabet installation directory
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see <u>Standard Logging for Alfabet Batch Utilities</u>.
Command line help	Start executable with `-h` or `-help`

The executable must be started with the following parameters:

```
AlfaAdministratorConsole.exe-msalias <alias name>-DbUser <username>-
DbPassword <password>-db_update <AMM file>
```

The table below displays the command line options:

Command Line Option	Mandatory/ Default	Explanation
`-db_update` `<AMM file>`	Mandatory	Update of the configuration in the target database must be defined with the parameter `-db_update` followed by the name of the AMM file

Command Line Option	Manda-tory/ Default	Explanation
		containing the configuration that shall be integrated into the Alfabet database on the database server.
-msalias <alias name>	Manda-tory	Enter the server alias name as specified in the `AlfabetMS.xml` config-uration file for access to the database.
-msaliasesfile <Alfabet configu-ration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
-DbUser <data-base user name>	Manda-tory	Database user name for access to the Alfabet database on the data-base server. (i) If the access mode to the Alfabet database is Windows Au-thentication, the command line option `-DbUser` must be de-fined without a value for the user name and `-DbPassword` is not required.
-DbPassword <database user password>	Optional	Password for access to the Alfabet database on the database server.

After applying a configuration to a database, it is recommended that you check the database for consistency with the meta-model.

Managing Assemblies

If a `DLL` file is developed by Software AG for your specific requirements, the file must be uploaded to the Alfabet database. This is typically done by a system administrator using the tool Alfabet Administrator. How upload is performed depends on the way the file is delivered by Software AG:

- When the assemblies are delivered as files, they are uploaded via the assembly management functionality of the Alfabet Administrator. For more information, see Uploading and Managing Assemblies with the Alfabet Administrator .

- When the assemblies are delivered in a `*.amm` file, the `*.amm` file must be applied to the target database with the **Update Meta-Model** functionality. For more information, see Uploading Assemblies via the *.amm Update File to Another Database.

It is recommended that assemblies are uploaded first to a test environment to test the effects on the existing database before they are implemented in the production environment. If the assembly impacts the solution configuration, it typically must first be uploaded to the database of a development environment and then reviewed in the test environment before being applied to the production environment.

The tool Alfabet Expand provides a mechanism to upload both the customized configuration and the assembly files to an `*.amm` updater file. This file can then be used to update the test and production environment to include both the `DLL` files and the required configuration changes. The `*.amm` file can also trigger the execution of scripts during the upload of assemblies, if required.

It is recommended that you test the impact of assembly upload to the production database via the mechanisms provided by Software AG in a staging environment before migrating the new features to the production database.
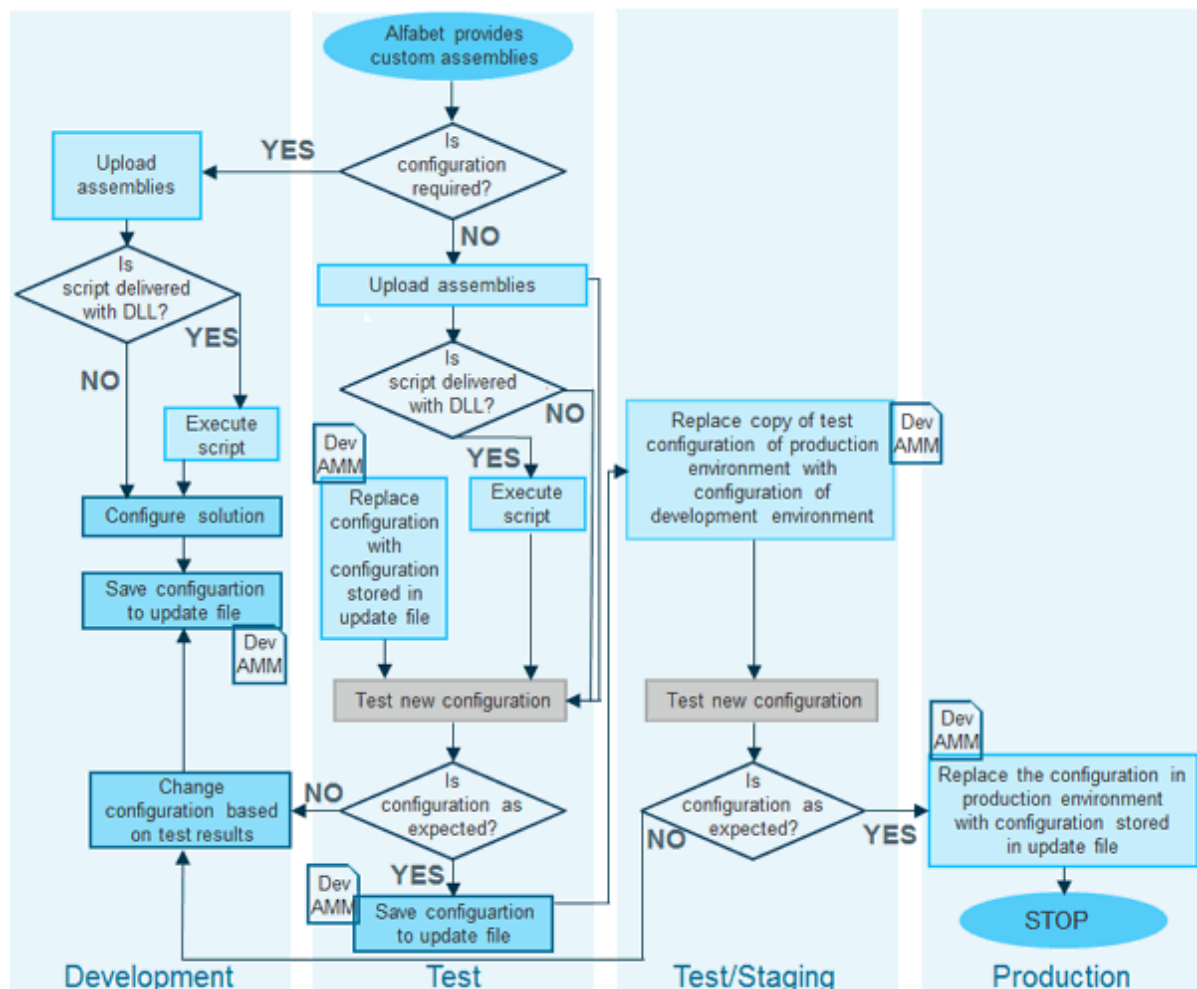

FIGURE: Example for a best practice workflow for the upload of custom assemblies to the database

Uploading and Managing Assemblies with the Alfabet Administrator

If a `DLL` is developed by Software AG for your specific requirements is delivered as a `*.dll` file, the file must be uploaded to the Alfabet database. This can be done in the **Assemblies** functionality of the Alfabet Administrator. This functionality also allows the deletion of assemblies from the database and provides an

overview of all assemblies currently uploaded to the database. The main Alfabet assembly uploaded by re-storing the database from an ADBZ file and by upgrading to a new release is also listed in the overview.

> ⓘ   `DLLs` may alternatively be delivered in an `*.amm` file. Upload is then performed as described in the section <u>Uploading Assemblies via the *.amm Update File to Another Database</u>. Once the assembly is uploaded, it is visible in the **Assemblies** functionality and can be managed there.

To upload an assembly to Alfabet:

1) In the **Alfabet Aliases** section of the **Administrator** explorer, right-click the server alias of the Alfabet Server connecting to the database that contains the configuration that you want to merge and select **Connect**. A login window is displayed.

2) Enter the user name and password for connection to the Alfabet database or, to connect via Windows sign-on, leave the fields empty and click **OK** to connect to the Alfabet database.

3) In the explorer, select **Assemblies**.

4) In the toolbar above the view on the right, click the **Upload Assembly** button.

5) Select the `DLL` file that you want to upload and click **Open**.

The `DLL` is now uploaded to the database. To delete the `DLL` at a later time, use the **Delete** button in the toolbar.

Executing a Script

Customer-specific assemblies delivered by Software AG may include scripts that must be executed to provide the functionality of the assembly.

To execute a script, follow the instructions provided by Software AG Support.

> ⚠   The execution of scripts is limited to scripts delivered by Software AG Support for various tasks such as maintenance procedures. Executing scripts that have not been provided by Software AG can cause severe damage to your Alfabet database.

Uploading Assemblies via the *.amm Update File to Another Database

The `*.amm` based update mechanism that allows you to update the configuration of a Alfabet database with the configuration of another Alfabet database can also be used to streamline the upload of assemblies to a production database.

After uploading the assemblies delivered by Software AG to a database, and optionally perform the required configurations for the functionality triggered by the assembly, you can create a `*.amm` file that contains the DLL files and can optionally trigger the following:

* the execution of scripts

* the merge or replace of the configuration of the target database with a configuration optimized for use of the assembly.

The *.amm file can then be used to upload the assemblies to a target database together with all scripting and configuration changes required for the assembly.

To save assemblies and optionally custom configuration to a *.amm file.

1) In the menu of Alfabet Expand, select **Meta-Model** > **Create Configuration Meta-Model Update File**. An editor window opens:

2) Click the **Browse** button next to the **Select Output File** field and select the location and file name for storing the *.amm file in the browser that opens.

3) Provide a meaningful name and description for the update performed by the *.amm file when applied to the target database in the **Name** and **Description** fields in the section **General**. The name and description are displayed in the **Update Meta-Model** dialog box when the *.amm file is used to update the configuration in a database. It allows the person performing the update to be informed about the content of the configuration in the *.amm file.

4) In the section **Assemblies**, all assemblies in the current database are listed in the **Add from Database** table. Click the **Upload** field of the assembly that you want to include in the *.amm file. An x is then displayed in the field:

Add from Database		
Name	Last Modified	Upload
ITPlan.dll	27/01/2011	x

☐ Remove All Assemblies from target Database before Updating

You can click the field again to deselect the assembly.

5) In the section **Assemblies**, select the following options related to the upload of the assembly, if applicable for your configuration:

- **Remove All Assemblies from target Database before Updating**: Select the checkbox if you want all assemblies of the target database to be deleted before upload of the assemblies in the *.amm update file.

  ⚠ This option deletes the assembly store of the target database. It should only be selected if all assemblies that are required to operate the target database are included in the *.amm update file.

- **Update Script**: This section allows the execution of a script during upload of an assembly to be defined. Instructions on how to fill in the fields in the section are provided by Software AG on delivery of the assembly.

  ⚠ The execution of scripts is limited to scripts delivered by Software AG Support for various tasks such as maintenance procedures. Executing scripts that have not been provided by Software AG can cause severe damage to your Alfabet database.

- **Description**: The information entered in the **Description** field

6) Optionally, you can select all or part of the configuration of the current database to be uploaded to the *.amm file together with the assembly by selecting the affected parts of the configuration in the sections **Meta-model Content** and **Guide Pages**.

If you want to upload only single configured elements (for example, only one workflow except all of the workflows), you can define the `*.amm` file with the **Find Meta-Model Objects for Deployment** functionality instead of using the **Create Configuration Meta-Model Update File** option to define the `*.amm` file.

To upload the assemblies stored in an `*.amm` update file to a target database, follow the procedure described in the section <u>Uploading Assemblies via the *.amm Update File to Another Database</u>.

Upgrading to a New or Patch Release of Alfabet

Software AG provides a new release of Alfabet every six months. Patch releases are distributed between regular releases in order to provide different language versions of Alfabet as well as resolve errors.

Migration of the current Alfabet database to the new release or patch release is required for each Alfabet upgrade.

⚠️ The patch level number of the Alfabet database and all client and server components of the Alfabet platform must match to work with Alfabet. You must update all components of the Alfabet platform to work with the new release or patch release. Alfabet Expand tools with a deviating patch number will not be able to connect to the Alfabet database.

General Procedure for Upgrades

The following process applies to updates from release 10.9.x to 10.11.x. The required procedures are described in detail in the following sections. The update procedures for each release may differ from each other. Please consult the documentations of the involved release levels.

Step Performed	Procedure	Description
	Evaluate system parameters before starting the update.	
	Configure the Alfabet Web Application to display a system shutdown message instead of the standard HTML start page that provides access to the Alfabet interface.	
	If applicable, disable all scheduled tasks concerning the system.	
	If applicable, shutdown the Alfabet Server Service.	
	If applicable, uninstall the Alfabet Server Service.	
	Backup the system.	
	Install the new version or patch release of Alfabet.	
	Migrate the database to the current metamodel.	

Step Per- formed	Procedure	De- scrip- tion
	If applicable, reinstall the Alfabet Server Service.	
	If applicable, restart the Alfabet Server Service.	
	Change the web.config files of the Alfabet Web Application to correspond to the changes in functionality for the new Alfabet release.	
	Configure the Alfabet Web Application to access Alfabet instead of displaying the shutdown message and reset the application pool of the Alfabet Web Application	
	Test the availability and functionality of the new release.	
	If applicable, enable all scheduled tasks concerning the system.	

Updates to an Alfabet release from a release prior to Alfabet 10.9 require a stepwise update.

For example, to update from release 10.7 to release 10.11:

- Update from release 10.7 to release 10.9 (there is no release 10.8)

- Update from release 10.9 to release 10.11 (there is no release 10.10)

The update procedure can differ from one release to the other. Consult the documentation of the respective release for a documentation of the required update processes.

Step 1: Evaluating the System Parameters

The update procedures described in the following require system parameter values as input. It is highly recommended that you evaluate the relevant system parameters before you upgrade to a new version of Alfabet in order to ensure that the settings defined for your system are recovered for the new version.

Each of the following procedure descriptions starts with a table listing all relevant system parameters with a column to enter the value required for your system. The system parameters are numbered, and the numbers are used in the descriptions of command line calls and procedures to specify which value must be entered in the respective code.

The following table lists all parameters that are relevant for any of the following step descriptions. It is an aggregation of the content of all parameter tables in the procedure descriptions:

Number	Parameter	Value

Alfabet Server related

Number	Parameter	Value
1	Alfabet Server Host	
2	Alfabet Server service executable file name	
3	Server Alias used for the Alfabet Server service	
4	Windows service name in the registry for the Alfabet Server service	
5	Display name of the Alfabet Server service in the Microsoft service Manager	
6	Start Mode (manual/automatic)	

If the Alfabet Server is located on a separate host only

Number	Parameter	Value
7	Path to the installation directory of the Alfabet components	
8	Installation mode (standard/custom)	
9	If installation mode is custom: Components to be installed on server side	
10	Path to the AlfabetMS.xml configuration file	
11	Folder name for the Alfabet components in the Windows Start Menu	
12	Path to the directory containing the search index files for Alfabet	

Web Server related (to be provided for each relevant Alfabet Web Application)

Number	Parameter	Value
13	Web Server Host	

Num-ber	Parameter	Value
14	Application Directory for the Alfabet Web Application	
15	Application Pool used for the Alfabet Web Application	
16	Path to the physical directory of the Alfabet Web Application	
17	Path to the HTML page displaying the outage message	
18	Path to the AlfabetMS.xml configuration file	
19	Path to the installation directory of the Alfabet components	
20	Folder name for the Alfabet components in the Windows Start Menu	
21	Installation mode (standard/custom)	
22	If installation mode is custom: Components to be installed on server side	
23	Path to the directory containing the search index files for Alfabet	
24	Remote Alias for connection of the Alfabet Web Application to the Alfabet Server (Only if the Alfabet Web Application connects to the Alfabet Server to send emails or to manage the full-text search functionality)	

Database related

24	Database Server Name	
25	Database Name	
26	Server Alias to connect to the Alfabet database	
27	Access Mode	

Num-ber	Parameter	Value
28	User Name for Database Access	
29	User Password for Database Access	
30	Path to the AMM file for migration of the database	

Client related

32	URL of the Alfabet Web Application	

Step 2: Configuring the Alfabet Web Application to Display a System Shutdown Message

This step is an optional but useful mechanism to inform users about current maintenance work on the Alfabet system and prevent complaints about an unexpected server shutdown.

During a planned system outage, you can temporarily substitute the standard HTML start page with another file by means of the Internet Information Services® Manager on the Web server that hosts the Alfabet Web Application. The standard HTML start page to access Alfabet via the Alfabet Web Application is the Home.aspx file located directly in the sub-directory of the Alfabet Web Application, which is located in the root directory of the Alfabet installation.

You can also configure a broadcast message to inform all system users who log in to the Alfabet Web Application about the upcoming planned shutdown. All broadcast messages that have been activated will be displayed in the banner at the bottom of the Alfabet screen. Broadcast messages are configured in the **Broadcast Messages** functionality available in the Alfabet user interface via an administrative user profile. For more information, see the section *Defining Broadcast Messages for the User Community* in the reference manual *User and Solution Administration*.

The following information is relevant for this procedure:

Number	Parameter	Value
13	Web Server Host	
14	Application directory for the Alfabet Web Application	

Number	Parameter	Value
16	Path to the physical directory of the Alfabet Web Application	
17	Path to the HTML page displaying the outage message	

To substitute the standard start file with another file providing inforamtion about the current system outage, do the following in the Internet Information Services® Manager on the Web Server host (13):

In Internet Information Services® 8:

1) Create an HTML page displaying the outage message (17) and store it in the sub-directory of the Alfabet Web Application in the root directory of the Alfabet installation (16).

2) In the explorer tree of the Internet Information Services® Manager, locate the application directory of the Alfabet Web Application (14).

3) In the **Features View**, double-click **Default Document** in the section **IIS**.

4) In the **Default Document** page, remove the standard document `Home.aspx` and any other configured standard documents.

5) In the **Actions** pane, click **Add** and enter the name of the HTML file containing the system shutdown message.

6) Close the Internet Information Services® Manager.

Step 3: Shutting Down Scheduled Tasks and Database Replication Mechanisms

Software AG provides several batch utilities that must be executed to activate specific Alfabet functionalities, to enhance the usability of some Alfabet functionalities, or to update data in the Alfabet database.

Executables for batch jobs can be started in a command line or by means of a Windows® batch job. When executing a Windows batch job, the execution time for a batch job is defined with the help of the Windows scheduler for batch jobs.

Prior to shutting down the Alfabet Server Service, all Windows® batch jobs running Alfabet batch utilities must be set to disabled.

Windows® batch job	Running the following batch utilities	Required shut-down window	Service Shut down performed at	Service restart performed at

The following table can be used as a support to plan and execute the required action:

Multiple interfaces for external applications are available for the Alfabet components (for example, Web Services and interfacing with external reporting tools). If your existing Alfabet application is interfacing with third-party components. If external applications rely on the availability of Alfabet, it might be required to provide out of service messages or stop the applications.

Please note that database replication mechanisms targeting the Alfabet database must be shut down during the upgrade procedure.

The following table can be used as a support to plan and execute the required action:

Affected Application	Interface with Alfabet	Action required during/prior to update period	Performed at	Action required after update period	Performed at

Step 4: Shutting Down the Alfabet Server Service

The following information is relevant for this procedure:

Number	Parameter	Value
1	Alfabet Server Host	
5	Display name of the Alfabet Server service in the Microsoft service Manager	

To shut down the Alfabet Server service:

1) On the Web Server host (1), click the **Start** [ ] icon that appears when you move the mouse to the lower left corner and click the **Server Manager** [ ] icon to open the Server Manager.

2) In the menu on the upper right of the Server Manager, select **Tools** > **Services**.

3) In the Services window, select the Alfabet Server service (5) in the list of services and click **Stop the service** on the left of the list.

Step 5: Uninstalling the Alfabet Server Service

The following information is relevant for this procedure:

Number	Parameter	Value
1	Alfabet Server Host	
2	Alfabet Server Service executable file name	

To uninstall the Alfabet Server Service:

1) On the Alfabet Server host, click the **Start** icon that appears when you move the mouse to the

   lower left corner and click the **Windows Power Shell** icon to open the Windows Power Shell.

2) Run the InstallUtil of the.NET framework on the alfaServerService.exe to uninstall the service. The following examples use the Microsoft® standard path to the utility.

   on a 32 bit operating system:

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\Installutil.exe-u
< Alfabet ServerService executable file name(2)>
```

   on a 64-bit operating system:

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Installutil.exe-u
<Alfabet Server Service executable file name(2)>
```

Step 6: Backing Up the Alfabet components and the Alfabet Database

The following information is relevant for this procedure:

Number	Parameter	Value
13	Web Server Host	
19	Path to the installation directory of the Alfabet components	
16	Path to the physical directory of the Alfabet Web Application	
24	Database Server Name	

Number	Parameter	Value
25	Database Name	

If the Alfabet Server is installed and located on a different host

1	Alfabet Server Host	
7	Path to the installation directory of the Alfabet components	

Backup of the database is done by using the backup mechanism on the database server level.

Backup of the Alfabet components is done by copying the content of the installation directory of the Alfabet components to another directory.

Component	Backup Located at	Backup Performed at
Installation directory of the Alfabet components components (19)		
Physical directory of the Alfabet Web Application (16)		
Alfabet database (24/25) backup file		

If the Alfabet Server is installed and located on a different host:

Path to the installation directory of the Alfabet components (7)		

Step 7: Uninstalling the Previous Release

After having created a backup of the Alfabet installation, the previous installation must be uninstalled.

The following information is relevant for this procedure:

Number	Parameter	Value
13	Web Server host	
19	Path to the installation directory of the Alfabet components	
16	Path to the physical directory of the Alfabet Web Application	

If the Alfabet Server is located on a separate host:

Number	Parameter	Value
13	Alfabet Server Host	
7	Path to the installation directory of the Alfabet components	

1) On the Web Server host, click the **Start** icon that appears when you move the mouse to the
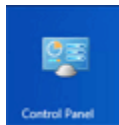
lower-left corner and click the **Control Panel** icon to open the Control Panel.

2) Click on **Uninstall a program**. A list of installed programs is displayed.

3) In the list, select your Alfabet installation and click **Uninstall** in the toolbar.

After having uninstalled the Alfabet installation, control the installation directory of the Alfabet components (19 / 16) and remove any remaining files.

If the Alfabet Server is located on a separate host, repeat the uninstallation for the Alfabet Server on the server host.

Step 8: Installing the New Release

There are two means to deliver patch releases or new releases. You will either be provided with a ZIP file Software AG or with an installation CD to install the new release. It is recommended that you clarify any special requirements for your installation prior to starting the upgrade process.

The following information is relevant for this procedure:

Number	Parameter	Value
13	Web Server host	

Number	Parameter	Value
14	Application Directory for the Alfabet Web Application	
19	Path to the installation directory of the Alfabet components	
21	Installation mode (standard/custom)	
22	If installation mode is custom: Components to be installed on server side	
23	Path to the directory containing the search index files for Alfabet	
20	Folder name for the Alfabet components in the Windows Start Menu	
16	Path to the physical directory of the Alfabet Web Application	
18	Path to the AlfabetMS.xml configuration file	

If the Alfabet Server is located on a separate host:

Number	Parameter	Value
13	Alfabet Server Host	
7	Path to the installation directory of the Alfabet components	
11	Folder name for the Alfabet components in the Windows Start Menu	
7	Installation mode (standard/custom)	
9	If installation mode is custom: Components to be installed on server side	
10	Path to the AlfabetMS.xml configuration file	

- You must have administrative rights in order to install Alfabet.

- The definition of the parameters for the upgrade must be identical to the previous installation. You must specify the following parameters for the installation:

  - The location and name of the root installation directory. For an upgrade to a patch release, the old installation can be overwritten. For the installation of a new release, the installation directory must be empty. In both cases a backup of the old version is required.

  - The components installed in the previous release (including the serial number type used for installation and selection of components chosen for a custom installation).

  - The location and name of the directory containing the files of the Alfabet Web Application.

  - The location and name of the Index directory containing the index files for Alfabet 's full-text search.

  - The name of the Alfabet directory in the Windows Start\Programs directory that is displayed in the Windows® menu **Start** > **Programs.**

If the Alfabet Server is located on the same host as the Alfabet Web Application, both components are installed in one installation procedure.

If the Alfabet Server is located on a separate host, the installation procedure must be run on both hosts (1)(13) installing the respective components.

The following workflow describes the complete installation of both components. If you install only a subset of the components on a host, some of the steps are skipped.

1) Insert the Alfabet CD or extract the installer files from the zip file and start the installation via auto run or launch the setup.exe file.

2) The **Install Shield Wizard** opens.
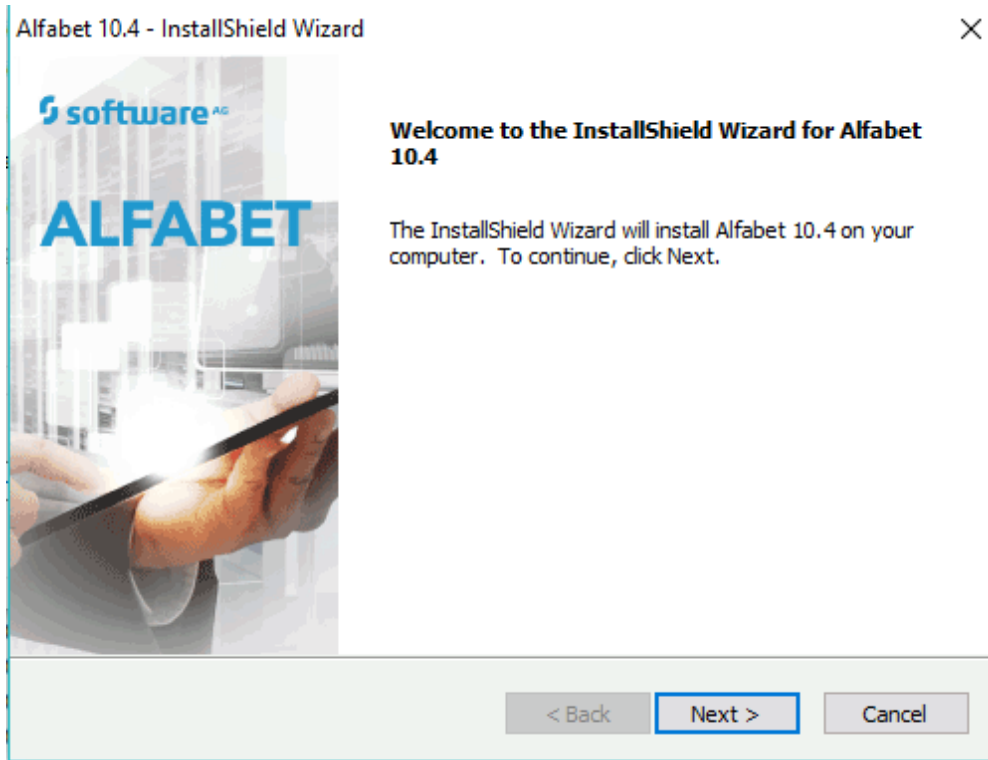
3) Click **Next** to continue.
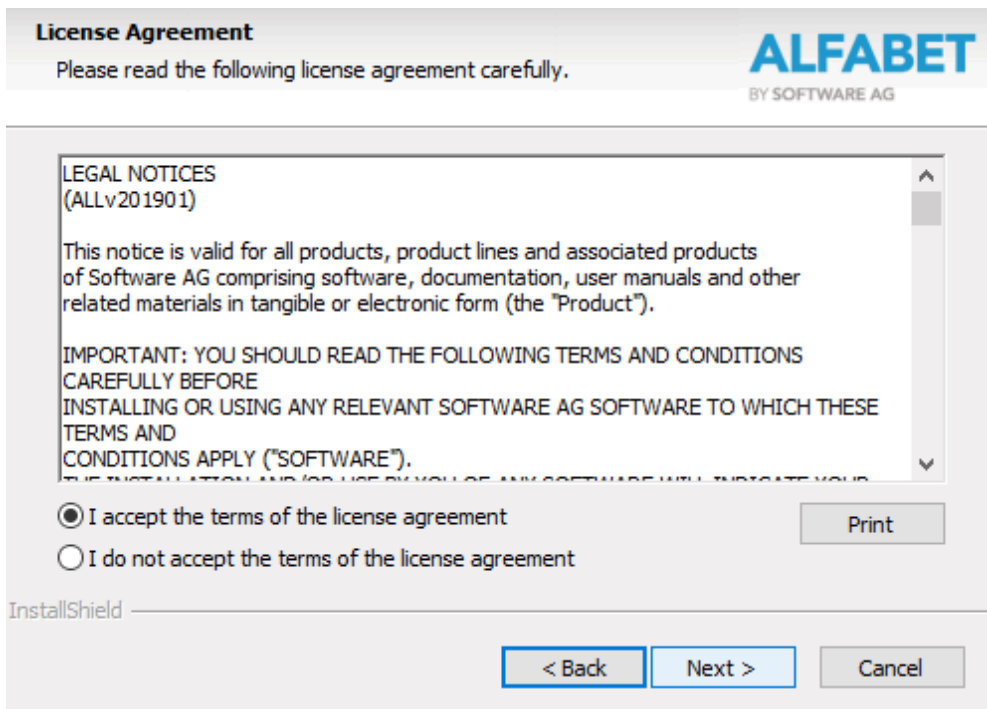
4) Select the checkbox of the option **I accept the terms of the license agreement** and click **Next** to continue.

5) Enter a user name, company name and the serial number delivered by Software AG. The serial number determines which features are available for installation.
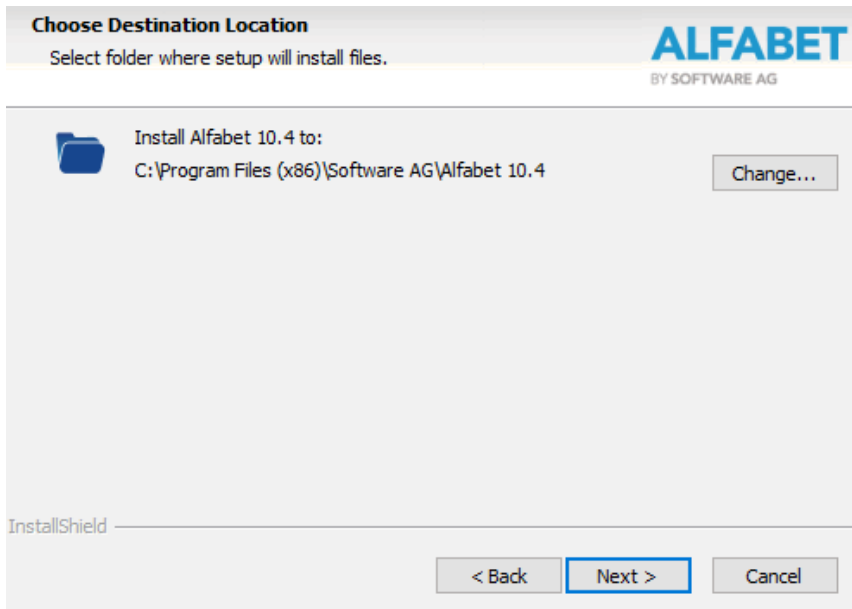
> ⓘ For information about serial numbers, see the section Scope of Delivery.

6) Click **Next** to continue.

7) Select the installation mode (21) and click **Next** to continue.

- **Standard** installs all available features. This is recommended for the installation of all Alfabet components included in your license.

- **Custom** allows you to choose features to install. It is recommended that you select this option even if all components will be installed. This allows you to review whether the installation includes all components that are required.

8) The default installation folder is displayed. If you want to install the Alfabet components in another location to make it identical to the installation folder used for the last release (19), click the **Change** button and select a destination folder for installation. If the destination folder does not exist, it will be created.

9) Click **Next** to continue.
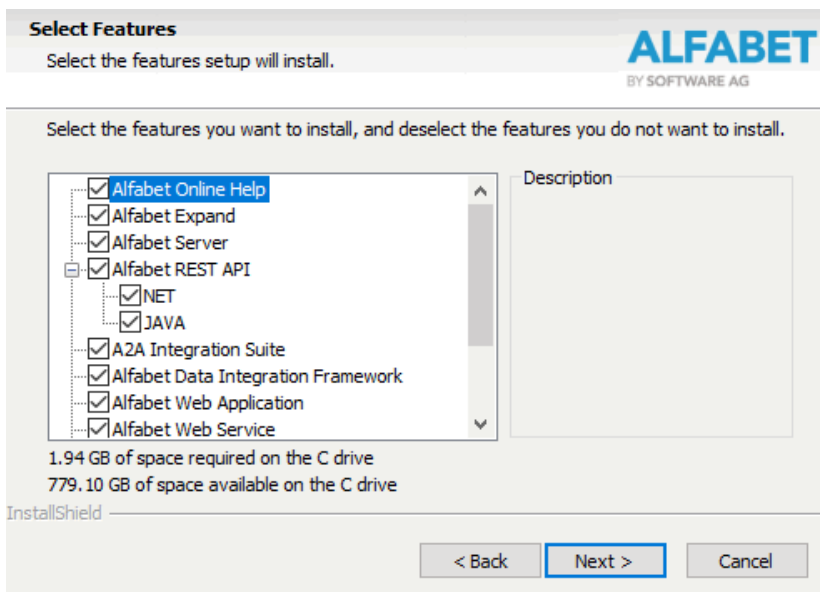
10) If the installation mode is **Custom**, control whether all required components (22) are checked. Select or deselect components, as needed.

> ⓘ The Alfabet Online Help is an integral part of the Alfabet Web Application and cannotbe deselected if the Alfabet Web Application is selected for installation. However, you can install the Alfabet Online Help separately with access via a Web server without installing the Alfabet Web Application.
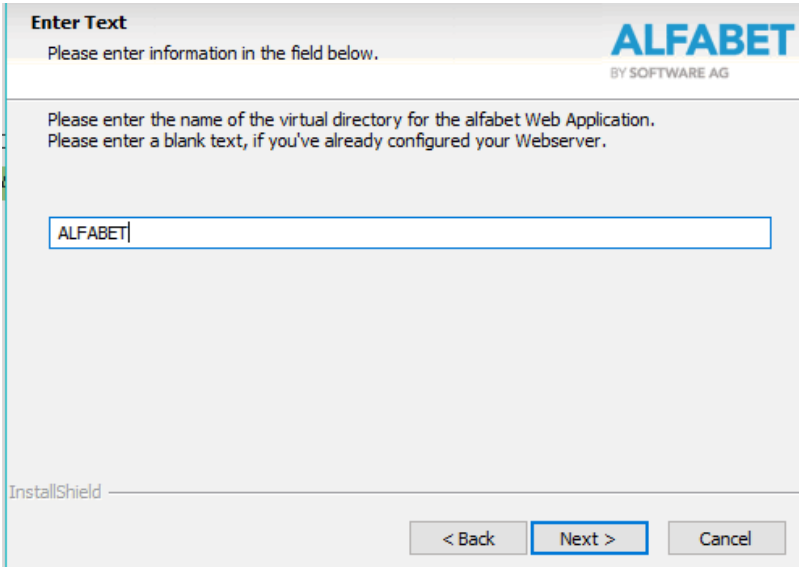
11) Click **Next** to continue.

12) Select the installation folder of the Alfabet Web Application files (`16`).

> ⓘ The Alfabet Online Help files are located in the **Help** subdirectory of the Alfabet Web Application directory.

13) Click **Next** to continue.

14) To use the application directory that was already used for the old installation (`14`), leave the field blank. Click **Next** to continue.

ⓘ To create the application directory with the setup procedure:

- Enter a directory name in the field of the installation shield. A directory with the specified name will be created in the Internet Information Services® during installation of the Alfabet components. The default value is Alfabet. Click **Next** to continue.

- After installation, you must configure the server alias of the Alfabet Web Application to use the application directory. For information about configuring the access to the application directory, see the section Creating a Server Alias for the Alfabet Web Application.

15) Click **Next** to continue.

16) Select the directory to install the index files required for the full-text search functionality available for Alfabet (23). The search index for the Alfabet online Help is created during installation in the **HelpSearchIndex** subdirectory of the selected index directory. Additional search indexes generated while working with Alfabet will be located in other sub-directories in the selected directory. After installation, you must configure the Alfabet Server to access this folder to find the search index files.
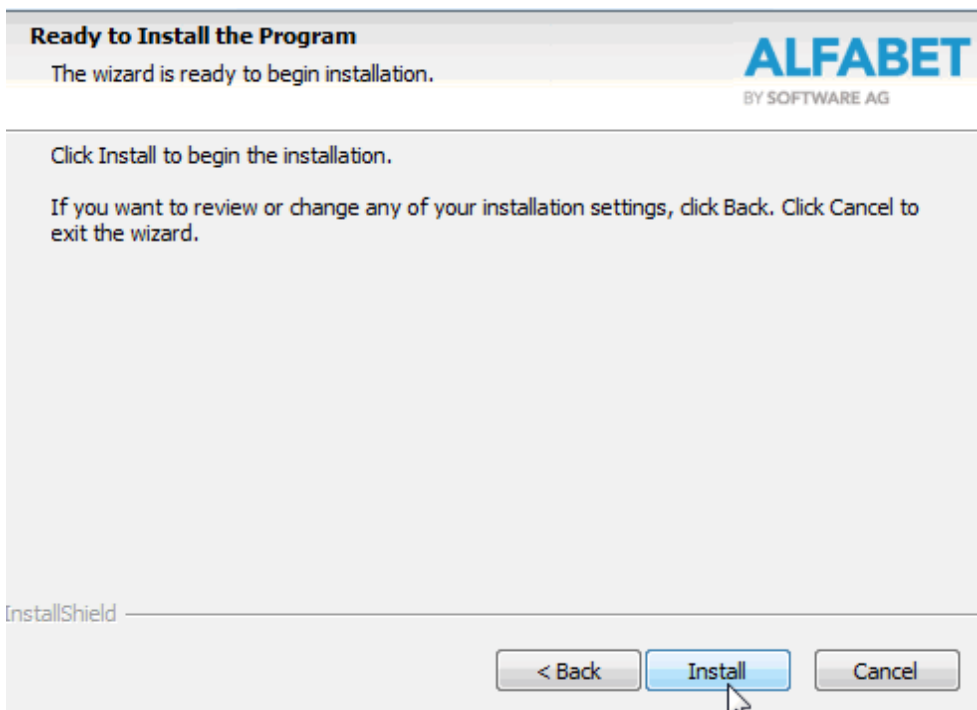
ⓘ For information about configuring the access to the search index files, see the section Creating a Server Alias for the Alfabet Web Application.
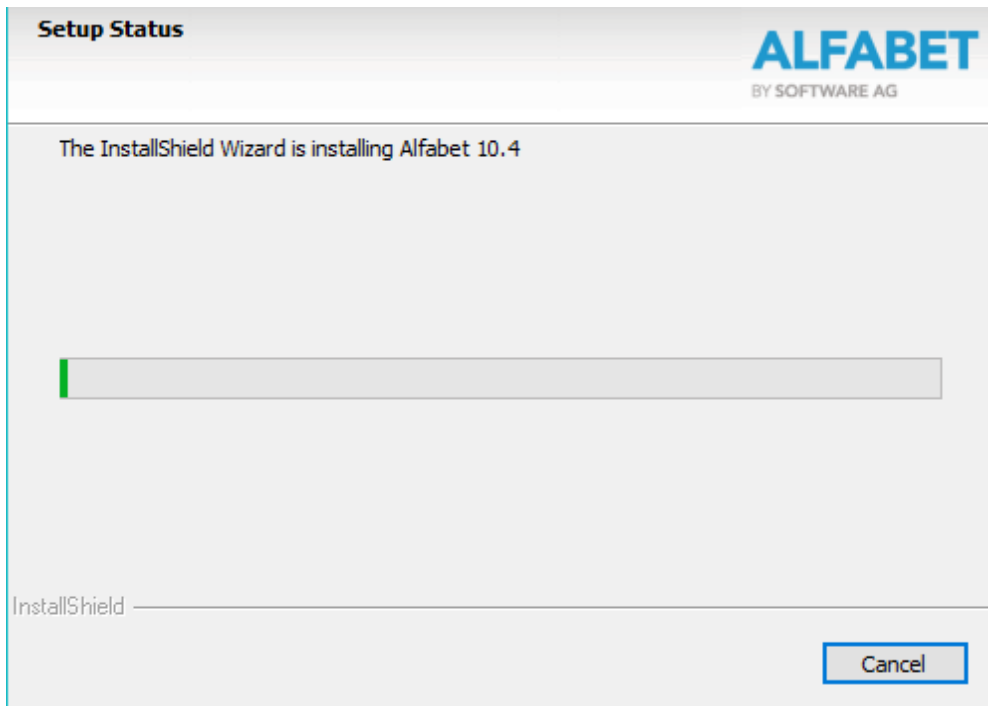
17) Click **Next** to continue.
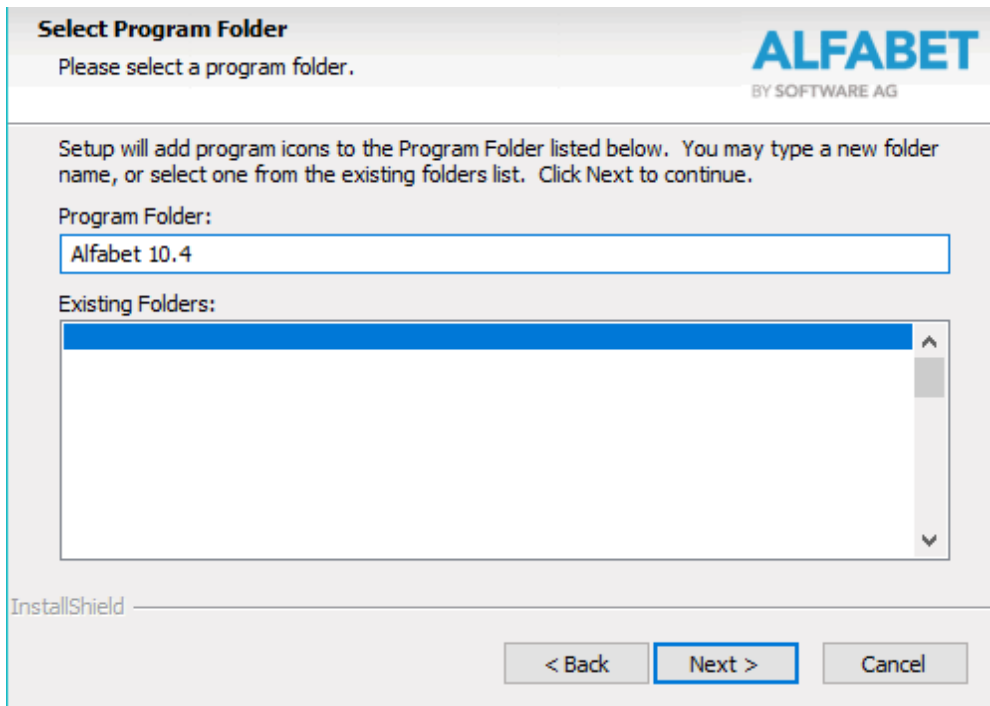
18) Enter the name of the folder where Alfabet will be located (20). This will be displayed in the
    Windows® **Apps** window as section heading.

19) Click **Next** to continue.
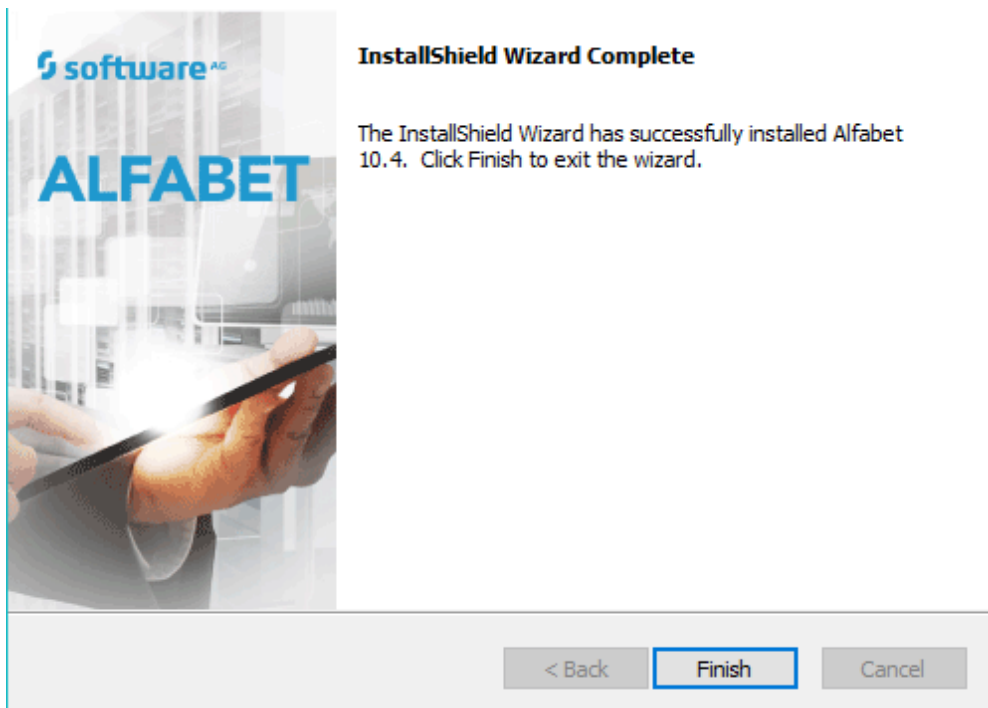

20) Click **Install** to start the installation.

21) The progress of the installation procedure is displayed.

 (i) The installation of the Alfabet Online Help may take some time.

22) After installation, additional required files are created and folder permissions are set.

After installation, the following window of the installation shield wizard will be displayed:

23) Click **Finish** to exit the wizard.

24) After installation, copy the `AlfabetMS.xml` file from your backup of the previous release to the corresponding folder of the new release (**18** / **10**).

Step 9: Migrating the Alfabet Database to the Meta-Model of the New Alfabet Release

The required changes to the meta-model for upgrade to the new release are stored in a.amm file delivered with the new release. Update of the meta-model in your database can be performed via a command line tool provided by Software AG.

> (i) If you have implemented the AlfaBot with the required connection to a DialogFlow® account, changes to intents and entities provided with the new release will be automatically updated in DiaglogFlow®. Make sure that a connection to your DialogFlow® account can be established during the update meta-model operation. Success of the operation will be logged in the update meta-model log file.
>
> For information about the AlfaBot capability and the configuration of the connection to the DialogFlow® NLP provider, see *Implementing the AlfaBot for Navigation via a Full-Text Command* in the reference manual *Configuring Alfabet with Alfabet Expand*.

The following information is relevant for this procedure:

Number	Parameter	Value
13	Alfabet Web Application host	
19	Path to the installation directory of the Alfabet components	
18	Path to the AlfabetMS.xml configuration file	
26	Server Alias to connect to the Alfabet database	
27	Access mode for database access	
28	User Name for database Access	
29	User password for database access	
30	Path to the AMM file for migration of the database	

To update the meta-model:

1) On the Web Server host, click the **Start** icon that appears when you move the mouse to the lower left corner and click the **Windows Power Shell** icon to open the Windows Power Shell.

2) Go to the installation directory of the Alfabet components (2).

3) Run

```
AlfaAdministratorConsole.exe-msalias <alias name (26)> [-msaliasesfile
<Alfabet configuration file path (11)>]-DbUser <username (28)>-DbPassword
<password (29)>-db_update <AMM file (30)>
```

The table below displays the process relevant command line options. In addition, logging can be configured via the command line. For information about standard logging and the command line options, see Standard Logging for Alfabet Batch Utilities in the reference manual *System Administration*.

Command Line Option	Mandatory/ Default	Explanation
`-db_update <AMM file>`	Mandatory	The name and path to the *.amm file used for restore of the configuration in the target database (30) must be defined with the parameter-db_update. The *.amm file is delivered in the `Database` subfolder of the Alfabet installation folder.
`-msalias <alias name>`	Mandatory	Enter the server alias name (26) as specified in the AlfabetMS.xml configuration file for access to the database.
`-msaliasesfile <Alfabet configu­ration file path>`	Optional	If the AlfabetMS.xml configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the AlfabetMS.xml file must be specified with this parameter (18).
`-DbUser <user name>`	Mandatory	User name for access to the Alfabet database on the database server (28). ⓘ If the access mode to the Alfabet database (27) is Windows Authentication, the command line option-`DbUser` must be defined without a value for the user name and-`DbPassword` is not required.
`-DbPassword <user password>`	Optional	Password for access to the Alfabet database on the database server (29).

If issues occure during update of the meta-model via AMM file, a log file is created during update of the meta-model. The log file is written to the directory that contains the AMM file and has the name <Name of the AMM file>_<Time Stamp>.log. Consult the log file to view any issues that occurred during update.

If configurations not matching the meta-model after update are detected in the database, the update meta-model process will write the problems into a Microsoft® Excel® file. The file is written to the directory that contains the AMM file and has the name <Name of the AMM file>_<Time Stamp>.xls. Open the Excel file and check for issues that require correction. The file consists of the following sheets:

- ***Meta-Model Errors***: Lists incompatibilities with the updated meta-model as well as hints for improvements based on new features. For upgrade to Alfabet 10.6, the information includes a list of filter controls in configured reports and custom selectors that can be enhanced in usability by enabling the new feature for look-ahead typing. These are all **Edit Field** interface controls (**Type** =Edit) that have the **Value Type** attribute set to String and that are not ReadOnly. For information about how to configure a filter field to allow look-ahead typing, see *Configuring Edit Fields* in the section *Defining Filters for Configured Reports and Selectors* of the reference manual *Configuring Alfabet with Alfabet Expand*.

- ***Translation Errors***: Lists issues detected during vocabulary update, like for example original strings that exceed the maximum allowed character number of 600 characters. For information about vocabulary management in Alfabet, see the chapter *Localization and Multi-Language Support for the Alfabet Interface* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- ***Report Help Index***: Lists all configured reports for which a link to a customer defined help file is defined in the **Standard Context-Sensitive Help Index** attribute of the configured report. A separate **Custom Context-Sensitve Help URL** attribute has been introduced on the custom report views for the definition of a link to a custom help. Existing definitions of customer defined help in the defined in the **Standard Context-Sensitive Help Index** attribute of the configured report should be moved to the **Custom Context-Sensitve Help URL** attribute, because the **Standard Context-Sensitive Help Index** attribute of the configured report is used for automatically setting standard help links for some types of configured reports and existing links may be overwritten by standard help definitions if changes to the configuration occur. Please note that the URL of the custom help file must be entered without the prefix CUSTOM_HELP: in the **Custom Context-Sensitve Help URL** attribute of the custom report view.

- ***Wrong Presentation Objects***: Lists all configured reports that are having a wrong presentation object assignment. Due to an already solved problem with copy and paste interactions, presentation objects were not correctly assigned to the report view on copy and paste into a new report.

> ⚠ Please note that for some reports the presentation object assignment is following special rules. If a report is not listed in the ***Wrong Presentation Objects*** list, the presentation object assignment should not be changed even if it does not match the correct settings described below.

Do the following for all configured reports listed in the ***Wrong Presentation Objects*** list:

1) Open the Reports tab of Alfabet Expand.

2) In the explorer, right-click the configured report that you want to correct and select **Set Report State to 'Plan'** from the context menu.

3) In the explorer, expand the node of the configured report that you want to correct.

4) Expand the report view 🖼 node beneath the configured report.

5) Click the presentation object 🔲 node beneath the configured report view.

6) In the attribute window, copy the content of the **Name** attribute.

7) In the explorer, double-click the report view ▣ node.

8) In the report view editor that opens, click the **Presentation: <Name>** interface control available as part of the report view.

9) In the attribute window, paste the name of the presentation object to the **Source** attribute.

10) In the toolbar, click the **Save** 💾 button to save your changes.

- ***Indicators Computation Rules***: Lists all indicator computation rules that are no longer valid because of meta-model changes. For information about the configuration of computation rules, see *Specifying Computation Rules for Indicator Types* in the reference manual *Configuring Evaluation and Reference Data in Alfabet*.

- ***Queries***: Lists all queries defined in Alfabet configuration that are no longer valid because of meta-model changes.

- ***Meta-Model Object Names***: Lists all meta-model object names that are problematic because they are not following implemented rules like for example the maximum number of allowed characters or unique key violation.

- ***Class Settings Editor***: Lists all class settings with an incorrect editor configuration. For information about the configuration of class settings, see *Configuring Class Settings for Object Classes and Object Class Stereotypes* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- ***Custom Selector Errors***: Lists all custom selectors that are not correctly assigned in the configuration. If a custom selector is listed here, it is for selection of objects of an object class that cannot be meaningfully selected in the context the custom selector is implemented. For example, custom selectors are listed if they are defined to the class settings of an object class that is not specified in the **Selectable Classes** attribute of the class entry of the custom selector.

- ***User Profile Errors***: Lists all user profiles with a missing or incorrect view scheme configuration. User profiles are listed here if the **View Scheme** attribute is either empty or set to a view scheme that does not exist. The view scheme configuration is required to apply the correct class settings to the user profile, thus ensuring that the correct set of functionality like using the AlfaBot and assignment of chart views to tabular configured reports is applied to the object classes for the user profile.

After applying a configuration to a database, it is recommended that you check the database for consistency with the meta-model as described in the section *Saving the Configuration of the Alfabet Solution to an AMM File* of the reference manual *Configuring Alfabet with Alfabet Expand*.

Step 10: Reinstalling the Alfabet Server Service

The following information is relevant for this procedure:

Number	Parameter	Value
1	Alfabet Server host	
7	Path to the installation directory of the Alfabet components	
2	Alfabet Server service executable file name	
3	Server Alias used for the Alfabet Server service	
4	Windows service name in the registry for the Alfabet Server service	
5	Display name of the Alfabet Server service in the Microsoft service Manager	
6	Start Mode (manual/automatic)	

To reinstall the Alfabet Server Service:

1) On the Web Server host, click the **Start** icon that appears when you move the mouse to the

    lower left corner and click on the **Windows Power Shell** icon to open the Windows Power Shell.

2) Go to the **Programs** sub-directory of the Alfabet Server 's root installation directory (7).

3) Run

    ```
    AlfaServerServiceGenerator.exe-server <server_alias_name(3)>
    ```

This program will create an executable file that can be installed as a service. The resulting service name is case-sensitive.

> Note the following when generating a server service:
>
> - Whitespaces are not allowed in the command line options of the **AlfaServerServiceGenerator**.
>
> - To ensure that valid Windows® executable names are generated through this process, the following restriction applies:
>
>   - The value provided for the parameter-exe (or if this is not defined the server alias name in AlfabetMS.xml) must be alphanumeric.

The only exceptions are the special characters '_' and "-". The administrator will be prompted with an error message if an invalid file name results from the server service generation.

The following command line options can be used with this command:

Parameter	Meaning	Use	Default Value
`-server <server_alias_name (3)>`	Server alias in the file AlfabetMS.xml	Man-datory	
`-service <service_name_windows (4)>`	Windows® service name that is used in the registry	Op-tional	"AlfaSrv" + server_alias_name
`-displayservice<service_display_name (5)>`	Displayed name in the Microsoft® Service Manager	Op-tional	"Alfabet Server Service " + server_alias_name
`-exe <service_exe_name (2)>`	Name of the executable file for the service	Op-tional	"AlfaServerService" + server_alias_name + ".exe"
`-start mode manual	automatic (6)`	Start type for the service	Op-tional	manual
`-h` `-help`	Displays the list of possible parameters	Alter-native	

4) Run the InstallUtil of the.NET framework on the alfa Server service executable. The following example uses the Microsoft® standard path to the utility on a 64-bit operating system:

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Installutil.exe
<Alfabet Server Service executable file name(2)>
```

The new service is now visible and can be started. It can also be further configured in the operating system console **Services**.

Step 11: Starting the Alfabet Server Service

• The Alfabet Server Service requires interaction with the desktop for technical reasons. This may lead to service shutdown when using a remote desktop session to the server host. If your Alfabet Server Service shuts down when disconnecting from a remote

desktop session, use Windows Remote Management or remote shell to start the Alfabet Server Service.

The following information is relevant for this procedure:

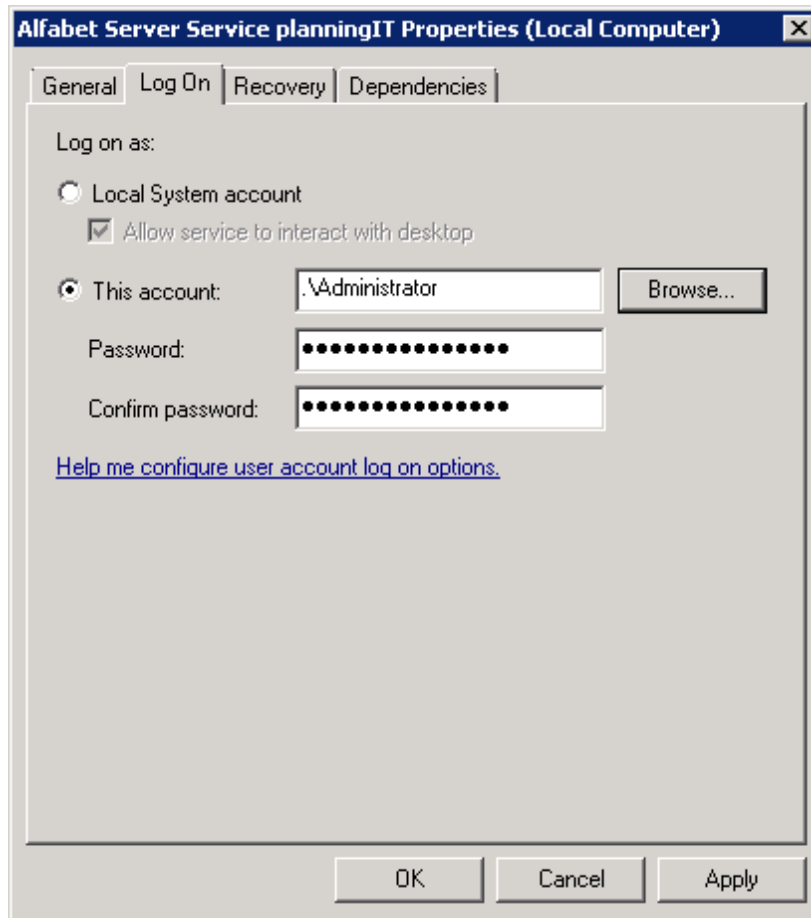| Number	Parameter	Value
1	Alfabet Server host	
5	Display name of the Alfabet Server service in the Microsoft service Manager	
5	Start Mode (manual/automatic)	

To start the Alfabet Server Service:

1) On the Web Server host, click the **Start** [icon] icon that appears when you move the mouse to the

   lower left corner and click on the **Server Manager** [icon] icon to open the Server Manager.

2) In the menu on the upper right of the Server Manager, select **Tools** > **Services**.

3) In the Services window, select the Alfabet Server service in the list of services and click **Stop the service** on the left of the list.

To start the Alfabet Server Service with each restart of the operating system, right-click the Alfabet Server Service in the list of services and select **Properties**. In the **General** tab of the editor that opens, set the **Startup Type** to **Automatic (Delayed Start)**.

If you are not a local administrator, open the **Log On** tab and change the **Log on as:** option to **This account:**. In the **This account:** field specify the account of the domain administrator with <domain>\<administrator> and enter the account's password in the **Password** and **Confirm Password** fields and click **Apply**:

Step 12: Reconfiguring the Alfabet Web Application web.config Files

It is recommended that you use the example `web.config` files delivered with the Alfabet Web Application that is optimized for use with the current release of Alfabet and adapt it to your system parameters.

Instead of one `web.config` file with all required settings, the `web.config` file is split up into three different files:

- `web.config` in the physical directory of the Alfabet Web Application. The XML elements `<alfaSection>` and `<appSettings>` have been removed from the file. Instead, links to two subordinate `web.config` files containing the settings for these XML elements have been added.

- `Alfabet.config` in the location defined in the link to the subordinate file in the `web.config` file. By default this is the `config` subdirectory of the Alfabet Web Application. This file contains the XML element `<alfaSection>` with all Alfabet related settings.

- `AppSettings.config` in the location defined in the link to the subordinate file in the `web.config` file. By default this is the `config` subdirectory of the Alfabet Web Application. This file contains the XML element `<appSettings>`.

Migration to the new structure is described in the following procedure.

The following information is relevant for this procedure:

Number	Parameter	Value
13	Web Server Host	
16	Path to the physical directory of the Alfabet Web Application	
18	Path to the AlfabetMS.xml configuration file	
26	Server Alias to connect to the Alfabet database	

In the ***Example*** subdirectory of the Alfabet Web Application, `web.config` file examples with up-to-date settings are available including a ***config*** sub-folder with the dependent files.

You can either use the example files or migrate your current web.config file to the current structure.

To use the example files, do the following::

1) On the Web server host (`13`), go to the physical directory of the Alfabet Web Application (`16`).

2) Copy the files `Alfabet.config` and `AppSettings.config` from ***Example/config*** to the folder ***config***.

3) Depending on the authentication method used, copy one of the following `web.config` files from ***Example*** folder to the main directory of the Alfabet Web Application and rename the file to `web.config`:

   - web.sso.saml2.config for SAML authentication

   - web.sso.windows.config for Windows authentication

   - web.default.config for all other authentication types

4) Open the `Alfabet.config` file in a text editor and change the path to the `AlfabetMS` file (`18`) and the relevant server alias name (`26`) in the following setting to match your installation and configuration:

   ```
   <add key="msfile" value="d:\path\to\Alfabetms.xml"/>
   <add key="alias" value="alias name"/>
   ```

To migrate your existing `web.config` file to the new structure:

1) On the Web server host (`13`), go to the physical directory of the Alfabet Web Application (`16`).

2) Copy the `Alfabet.config` and `AppSettings.config` files from ***Example/config*** to the folder ***config***.

3) Open the `web.config` file and the new files `Alfabet.config` and `AppSettings.config` in a text editor and perform the following changes:

   - Cut the entire XML element `<alfaSection>` from your `web.config` file and overwrite the XML element `<alfaSection>` in the `Alfabet.config` file with your `<alfaSection>`.

- Cut the entire XML element `<appSettings>` from your `web.config` file and overwrite the XML element `<appSettings>` in the `AppSettings.config` file with your `<appSettings>`.

- In the `web.config` file, add the following links in and after the XML element `<alfaGroup>`.

```
<alfaGroup>

<alfaSection configSource="config\Alfabet.config"/>

</alfaGroup>

<appSettings file="config\AppSettings.config">

</appSettings>

<location path="config">
```

Step 13: Reconfiguring the Alfabet Web Application to Access Alfabet

If you redirected the standard start page for the Alfabet Web Application to another HTML page with a shutdown message, you must now reset the original `Home.aspx` or `index.html` file of the Alfabet Web Application:

The following information is relevant for this procedure:

Number	Parameter	Value
13	Web Server Host	
14	Application Directory for the Alfabet Web Application	
15	Application Pool used for the Alfabet Web Application	

To substitute the standard start file with another file providing information about the current system outage, do the following in the Internet Information Services® Manager on the Web Server host (13):

In Internet Information Services® 8:

1) In the explorer tree of the Internet Information Services® Manager, locate the application directory of the Alfabet Web Application (14).

2) In the **Features View**, double-click **Default Document** in the section **IIS**.

3) In the Default Document page, remove the standard document displaying the outage message.

4) In the **Actions** pane, click **Add** and enter `Home.aspx` or `index.html`.

5) In the explorer tree of the Internet Information Services® Manager, locate the application pool of the Alfabet Web Application (15).

6) Reset the application pool.

7) Close the Internet Information Services® Manager.

Step 14: Test Connectivity and Functionality

You can test the connectivity of the Alfabet Web Clients as described below. After testing the connectivity, it is recommended that you test the performance of the new release including functionality tests that are typical for your enterprise's use of Alfabet.

The following information is relevant for this procedure:

Number	Parameter	Value
32	URL of the Alfabet Web Application	

1) Open a Web browser.

2) Enter the URL of the Alfabet Web Application (32)>.

3) Click **OK** to connect.

4) Login to Alfabet (if enterprise authentication is not used).

5) Select a user profile (if multiple profiles are attached to the user). The Alfabet user interface opens.

Step 15: Enable Scheduled Tasks

After successful update, set all Windows® batch jobs running Alfabet batch utilities to enabled and affected third party components can be configured to re-connect to Alfabet.

See Step 3: Shutting Down Scheduled Tasks and Database Replication Mechanisms for a list of the affected batch jobs and third party components.

Using the AlfaAdministratorConsole.exe for Automation of Maintenance Tasks

Software AG provides a command line tool for execution of the central actions required for database maintenance and upgrade to new releases. This makes it easier to automate maintenance tasks.

Executable	`AlfaAdministratorConsole.exe` located in the ***Programs*** subdirectory of the Alfabet installation directory
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see Standard Logging for Alfabet Batch Utilities.
Command line help	Start executable with `-h` or `-help`

The following sections give an overview over the actions that can be performed with the tool and the command line definition required for each action:

- Changing the Database Connection in the Server Alias Configuration

- Changing the Database User Name and Password in the Server Alias Configuration

- Changing the Server Variables in the Server Alias Configuration

    - Defining Server Variables via a Command Line Tool

    - Remote Setting of Server Variables With Encryption

- Updating a Database from an AMM File

- Creating an AMM File

- Archiving the Database in an ADBZ File

- Restoring the Database from an ADBZ File

- Triggering Database Accessibility Monitoring Events

- Triggering User Password Regeneration

- Resetting a User Password

- Anonymizing Data of Selected Users

- Anonymizing Object Data

- Releasing a Database Restricted Mode

- Rebuilding Indexes on Database Tables

- Updating Maintenance Window Definitions for the Job Schedule Functionality

Changing the Database Connection in the Server Alias Configuration

For each Alfabet component, the connection to the Alfabet database the component connects to is defined in the server alias configuration of the component in the **Database Settings** tab of the server alias configuration. The server alias configuration is stored in the file AlfabetMS.xml that must be located in the working directory of the component, or, for the Alfabet Web Application, at the location defined in the `Alfabet.config` configuration file of the Alfabet Web Application.

> ⓘ For an overview of the configuration files required for the Alfabet Web Application components, see Basic Configuration of the Alfabet Components.

For an existing server alias configuration, the database connection settings can be changed via `AlfaAdministratorConsole.exe` with the following command line:

```
AlfaAdministratorConsole.exe-msalias <alias name>-db_set_connection-DbDriver
<SqlServer|Oracle>-DbUser <database user name>-DBPassword <database user
password>-DbEncryptConnection <true|false>-DbCleanInvalidChars <true|false>-
DbDatabase <TNS connection name>
```

The table below displays the command line options:

Command Line Option	Mandatory/ Default	Explanation
`-db_set_con-nection`	Mandatory	To change the database connection information in the server alias configuration: `-db_set_connection`
`-msalias` <alias name>	Mandatory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.
`-msalias-esfile` <Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-DbDriver` <driver type>	Mandatory	Specify the database server type the Alfabet database is hosted on. Allowed values are `SqlServer` for a connection to a Microsoft® SQL Server® and `Oracle` for a connection to an Oracle® database server.
`-DbUser` <database user name>	Mandatory	Specify the user name for access to the Alfabet database on the database server. For Oracle® databases, the user name is identical to the schema name.

Command Line Option	Mandatory/ Default	Explanation
`-DbPassword` <database user password>	Mandatory	Specify the password for access to the Alfabet database on the database server.
`-DbEn- cryptConnec- tion` <true|false>	Optional	For a connection to a Microsoft® SQL Server® only: If the parameter is set to true, SSL/TLS will be applied to the connections between the Alfabet component and the Alfabet database. If the parameter is not defined in the command line, it is set to `false` in the server alias configuration.
`-DbClean- InvalidChars` <true|false>	Optional	Set this command line option to `true` if you want strings to be checked for validity prior to check-in or when reading the string from the Alfabet database. If the parameter is not defined in the command line, it is set to `false` in the server alias configuration.
`-DbDatabase` <database name>	Mandatory if `-DbDriver SqlServer` is set or if `-DbDriver Oracle-dbIn- directConnec- tion true` is set	Enter the name of the Alfabet database. Use the notation servername\databasename for a default instance or servername\instancename\databasename for a named instance.
`-DbIn- directCon- nection` <true|false>	Optional	For a connection to an Oracle® database only: Set to `true`, if an indicect connection to the Oracle® database should be established or to `false`, if a direct connection should be established. If the parameter is not defined in the command line, it is set to `false` in the server alias configuration. The direct connection is recommended for connections to the Alfabet database for performance reasons.
`-DbDatabase` <TNS connec- tion name>	Mandatory if either `-DbDriver SqlServer` or `-DbDriver Oracle-dbIn- directConnec- tion true` is set	Enter the name of the Alfabet database: • For a Microsoft® SQL Server® database, use the notation servername\databasename for a default instance or servername\instancename\databasename for a named instance. • For an indirect connection to an Oracle® database, use the Net Service Name of the database configured with the Oracle® Net Assistant. • For a direct connection to an Oracle® database this parameter is ignored.

Command Line Option	Mandatory/ Default	Explanation
`-DbHost` <database host name>	Mandatory if-`DbDriver Oracle-dbIndirectConnection false` is set	For a direct connection to an Oracle® database only. Specify the host name or IP address of the database server host.
`-DbPort` <port number>	Mandatory if-`DbDriver Oracle-dbIndirectConnection false` is set	For a direct connection to an Oracle® database only. Specify the port of the database server host to be used for connection to the Alfabet database.
`-DbService` <service name>	Mandatory if-`DbDriver Oracle-dbIndirectConnection false` is set	For a direct connection to an Oracle® database only. Specify the SID of the Alfabet database.

If the command is not correctly specified, the command line tool returns error messages in the command prompt.

Changing the Database User Name and Password in the Server Alias Configuration

For each Alfabet component, the connection to the Alfabet database the component connects to is defined in the server alias configuration of the component in the **Database Settings** tab of the server alias configuration. The server alias configuration is stored in the file AlfabetMS.xml that must be located in the working directory of the component, or, for the Alfabet Web Application, at the location defined in the `Alfabet.config` configuration file of the Alfabet Web Application.

> (i) For an overview of the configuration files required for the Alfabet Web Application components, see Basic Configuration of the Alfabet Components.

For an existing server alias configuration, the user name and password for the database connection settings can be changed via `AlfaAdministratorConsole.exe` with the following command line:

```
AlfaAdministratorConsole.exe-msalias <alias name>-ms_set_db_userpassword-
DbUser <database user name>-DBPassword <database user password>
```

The table below displays the command line options:

Command Line Option	Mandatory/ Default	Explanation
`-ms_set_db_userpassword`	Mandatory	To change the database user name and database user password in the database connection information in the server alias configuration: `-ms_set_db_userpassword`
`-msalias` <alias name>	Mandatory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.
`-msaliasesfile` <Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-DbUser` <database user name>	Mandatory	Specify the user name for access to the Alfabet database on the database server. For Oracle® databases, the user name is identical to the schema name.
`-DbPassword` <database user password>	Mandatory	Specify the password for access to the Alfabet database on the database server.

Changing the Server Variables in the Server Alias Configuration

The definition of server variables allows you to store information about connection strings to external sources in the server alias configuration. Storing the information about the connection strings in the server alias configuration instead of directly defining them in the configuration eases the propagation of changes.

Server variable names can be added to the following configurations. They will be substituted at runtime with the value of the server variable defined in the server alias configuration of the Alfabet component.

- connection strings to external data sources (for more information see Integrating Data from External Sources).

- URLs defined for dynamic Web links (for more information see the section *Configuring Dynamic Web Links That Users Can Access* in the reference manual *Configuring Alfabet with Alfabet Expand*).

- URLs defined for external reports (for more information see the section *Configuring Reports* in the reference manual *Configuring Alfabet with Alfabet Expand*).

- URLs defined for custom online help (for more information see the section *Providing Custom Online Help to the User Community* in the reference manual *Configuring Alfabet with Alfabet Expand*).

- Value definitions for XML attributes defining connections to external systems in the XML objects for the configuration of the interfaces to external systems. This applies to all XML objects located in Alfabet Expand in the **Presentation** tab in the subfolder **Integration Solutions** of the **XML Objects** folder. For more information, see the documentation of the respective interface.

Server variables can be set directly in the server alias editor, remotely via a command line tool with direct access to the server alias configuration file, or via an external editor that encrypts the server variables without direct access to the server alias configuration:

- [Defining Server Variables via a Command Line Tool](#)

- [Remote Setting of Server Variables with Encryption](#)

Defining Server Variables via a Command Line Tool

For an existing server alias configuration, a server variable can be changed or added via the command line tool `AlfaAdministratorConsole.exe` with the following command line:

```
AlfaAdministratorConsole.exe-msalias <alias name>-editvariables-variable
<variable name>-value <variable value>
```

For an existing server alias configuration, a server variable can be deleted via the command line tool `AlfaAdministratorConsole.exe` with the following command line:

```
AlfaAdministratorConsole.exe-msalias <alias name>-editvariables-variable
<variable name>
```

The table below displays the command line options:

Command Line Option	Manda-tory/ Default	Explanation
`-editvariables`	Manda-tory	To change the variable definitions in the server alias configuration, the command line must contain the parameter: `-editvariables`
`-msalias <alias name>`	Manda-tory	Enter the server alias name as specified in the `AlfabetMS.xml` config-uration file for access to the database.
`-msaliasesfile <Alfabet configu-ration file path>`	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-variable <vari-able name>`	Manda-tory	Specify the name of the server variable. If a server variable with the specified name does not exist in the server alias, a new server variable will be created.

Command Line Option	Mandatory/ Default	Explanation
		If a server variable with the specified name exists in the server alias, the value will be changed to the value defined with `-value`. To delete an existing server variable, specify the existing server variable name with `-variable` and do not add a `-value` definition to the command line. ⓘ The variable name can only contain letters (of the English alphabet), numbers and underscore.
`-value` <variable value>	Optional	Specify the value for the server variable. To delete an existing server variable, specify the existing server variable name with `-variable` and do not add a `-value` definition to the command line. ⓘ The following characters are not allowed in server variable values: " < > These characters can be written as HTML code in the server variable value: • `>` for > • `<` for < • `"` for "

Remote Setting of Server Variables with Encryption

Server variables can be provided encrypted in a separate file that can then be imported by a system administrator into a server alias configuration without knowledge about the provided server variable value. The server variable value will be displayed encrypted in the server alias editor.

To make encrypted editing of server variables available to other system administrators:

1) In the Alfabet Administrator, click the **Aliases** node.

2) Select the server alias that you want to define server variables for in the table and click the
   Edit 🖉 button.

3) In the server alias editor, open the **Variables** tab and click the **New** button.

4) Define a name for the server variable in the **Variable Name** field and a dummy value in the **Variable Value** field. The variable value will be overwritten later in the external editor. Repeat this for as much server variables as needed.

5) In the **Variables** tab of the server alias editor, select all server variables that shall be edited externally in the list of server variables, click the **Export** button select a file name and location and click **OK**. The output file must have the extension `.alfams`.

6) It is recommended that the command line tool `AlfaVariablesEditor.exe` is removed from the `Programs` folder and exclusively made available to persons which are responsible for adding the server variables to the `AlfabetMS.xml` file. The tool must be run in a folder containing the `*.alfams` file that shall be changed or read as well as `AlfaCore.dll` and `AlfaCommon.dll` files from the `Programs` folder of the Alfabet installation.

7) External administrators can run the tool `AlfaVariablesEditor.exe` with one of the following command lines to change the server variables:

- To list the encrypted server variables in an `AlfabetMS.xml` configuration file, start the command line tool with:

      AlfaVariablesEditor.exe-list <Path to the *.alfams server variable
      file>

- To add server variables encrypted to an `AlfabetMS.xml` configuration file, start the command line tool with:

      AlfaVariablesEditor.exe-update <Path to the *.alfams server variable
      file>-<variable name> <plain text variable value>

  Multiple variables can be defined with one command. To clear a value for a variable, specify:-`<variable name> null`

8) To import the server variables in the changed file, open the server alias editor for the server alias in the Alfabet Administrator and open the **Variables** tab.

9) Click the **Import** button, select the ***.alfams** file containing the data and click **OK**.

Variables that have been imported cannot be edited in the server alias any longer and the value is not displayed.

Updating a Database from an AMM File

Software AG provides a mechanism that allows the customer configuration to be saved and restored in another database independent of other parts of the database:

The customer configuration store and optionally the solution store can be saved to an AMM updater file. The updater file allows the configuration of a target database to either be merged or replaced by the configuration stored in the AMM updater file.

AMM updater files are also used during upgrades to a new Alfabet release to apply changes to the official metamodel to the customer database.

⚠ **Always back up the target database prior to updating the meta-model with configurations stored in an AMM file!** If the process fails, this may disrupt database integrity.

To update the meta-model with the configurations in an AMM file, use the following command line for the `AlfaAdministratorConsole.exe`:

    AlfaAdministratorConsole.exe-msalias <alias name>-DbUser <username>-
    DbPassword <password>-db_update <AMM file>

The table below displays the command line options:

Command Line Option	Manda-tory/ Default	Explanation
-db_update <AMM file>	Manda-tory	Update of the configuration in the target database must be defined with the parameter-db_update followed by the name of the AMM file containing the configuration that shall be integrated into the Alfabet database on the database server.
-msalias <alias name>	Manda-tory	Enter the server alias name as specified in the AlfabetMS.xml config-uration file for access to the database.
-msaliasesfile <Alfabet configu-ration file path>	Optional	If the AlfabetMS.xml configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the AlfabetMS.xml file must be specified with this parameter.
-DbUser <data-base user name>	Manda-tory	Database user name for access to the Alfabet database on the data-base server. ⓘ If the access mode to the Alfabet database is Windows Au-thentication, the command line option-DbUser must be de-fined without a value for the user name and-DbPassword is not required.
-DbPassword <database user password>	Optional	Password for access to the Alfabet database on the database server.

Creating an AMM File

Software AG provides a mechanism that allows the customer configuration to be saved and restored in an-other database independent of other parts of the database:

The customer configuration store and optionally the solution store can be saved to an AMM updater file. The updater file allows the configuration of a target database to either be merged or replaced by the con-figuration stored in the AMM updater file.

To store the complete meta-model configuration of the current database including all assemblies to an AMM file, use the following command line for the AlfaAdministratorConsole.exe:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <username>-
alfaLoginPassword <password>-mm_create_update-MmUpdateName <update caption
in AMM>-MmUpdateDescription <update description in AMM>-MmUpdateOutputFile
<AMM file Name>
```

To store all meta-model configuration objects tagged with defined tags to an AMM file, use the following command line for the `AlfaAdministratorConsole.exe`:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <username>-
alfaLoginPassword <password>-mm_create_update_by_tag-MmUpdateTags <tag
name>-MmUpdateName <update caption in AMM>-MmUpdateDescription <update
description in AMM>-MmUpdateOutputFile <AMM file Name>
```

The table below displays the command line options:

Command Line Option	Mandatory/Default	Explanation
`-mm_create_update`/`-mm_create_update_by_tag`	Mandatory	The command line must include one of the following parameters according to the kind of AMM file that shall be created: • `-mm_create_update` for the creation of an AMM file containing a complete configuration • `-mm_create_update_by_tag` for the creation of an AMM file containing tagged configuration objects only.
`-msalias` <alias name>	Mandatory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.
`-msaliasesfile` <Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName` <Alfabet user name>	Mandatory	Alfabet user name for login. ⓘ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPassword` <Alfabet user password>	Optional	Alfabet user password for login.
`-MmUpdateName` <update caption in AMM>	OptionalS	Specify a string that will be displayed as title of the meta-model update to the administrator performing a meta-model update on basis of this file.
`-MmUpdateDescription` <update description in AMM>	Optional	Specify a string that will be displayed as description of the meta-model update to the administrator performing a meta-model update on basis of this file.

Command Line Option	Mandatory/Default	Explanation
`-MmUpdateOutputFile <AMM file Name>`	Mandatory	Specify the path to and name of the AMM file to be created. The file must have the extension.amm.
`-MMUpdateTags <comma-separated tag list>`	Optional	To add only configuration objects that are tagged with defined configuration tags to the AMM file, specify the tag or tags with this parameter. If multiple tags shall be included, they must be specified as a comma-separated lists with no whitespaces between commas and tag names. ⓘ This parameter does not accept values defined in double quotes. A valid definition would be: `-MMUpdateTags tag1,tag2,tag3`

Archiving the Database in an ADBZ File

Alfabet databases can be archived in Alfabet database archive files (ADBZ) that are a proprietary file format. This archiving must not be used for normal regular database backup processes but rather, for example, to archive a database and send it to Software AG Support.

To archive the database in an ADBZ file, use the following command line for the `AlfaAdministratorConsole.exe`:

```
AlfaAdministratorConsole.exe-msalias <alias name>-DbUser <username>-
DbPassword <password>-db_archive-ArchiveUserSettings <false|true>-
ArchiveAudit <false|true>-ArchiveInternalDocs <false|true>-ArchiveOutputFile
<file name and location>
```

The table below displays the command line options:

Command Line Option	Mandatory/Default	Explanation
`-db_archive`	Mandatory	The command line must include the following command to archive the database to an ADBZ file: `-db_archive`
`-msalias <alias name>`	Mandatory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.

Command Line Option	Mandatory/ Default	Explanation
`-msaliasesfile` <Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-DbUser` <database user name>	Mandatory	Database user name for access to the Alfabet database on the database server. ⓘ If the access mode to the Alfabet database is Windows Authentication, the command line option `-DbUser` must be defined without a value for the user name and `-DbPassword` is not required.
`-DbPassword` <database user password>	Optional	Database user password for access to the Alfabet database on the database server.
`-ArchiveOutputFile` <file name and location>	Mandatory	Specify the path to and name of the ADBZ file to be created. The file must have the extension.adbz.
`-ArchiveUserSettings` <false|true>	Optional	To include the current user settings in the database in the ADBZ file, set this parameter to `true`.
`-ArchiveAudit` <false|true>	Optional	To include the audit history tables in the database in the ADBZ file, set this parameter to `true`.
`-SqueezeAudit` <false|true>	Optional	If set to `true`, the history tables in the database in the ADBZ file will be are scanned for obsolete entries. Any entries that were generated during, for example, the batch update of data via batch utilities without documenting any audit relevant changes are deleted from the audit tables stored in the ADBZ file.
`-ArchiveInternalDocs` <false|true>	Optional	To include the documents in the Internal Document Selector of the database in the ADBZ file, set this parameter to `true`.

Restoring the Database from an ADBZ File

Alfabet databases can be archived in Alfabet database archive files (ADBZ) that are a proprietary file format. This archiving must not be used for normal regular database backup processes but rather, for example, to archive a database and send it to Software AG Support.

⚠️ Please note the following:

- This action overwrites the content of the Alfabet database. **Never restore a database without prior backup of the current database! If the restore action fails, it may result in a corrupt database!**

- This mechanism is only available if the database server is installed on the same machine as the Alfabet Administrator. It is not intended for use in productive environments and should only be used to ease database maintenance in development or test environments.

- The restore of a database will fail if the server alias is configured to enforce password criteria and the database in the ADBZ file contains users with empty passwords or with a password not matching the defined user password criteria. If restore of the database fails, the target database may be corrupt.

- A case-sensitivity check has been implemented for restore of the Alfabet database from ADBZ file. If the target database is case-insensitive and the ADBZ file was created from a case-sensitive database, the restore process will not be performed and a message informs about the incompatibility of the databases.

- Prior to using this functionality, ensure that no Alfabet components are currently connected to the Alfabet database. This includes the Alfabet Web Application, the Alfabet Server (service), Alfabet Expand or batch utilities.

To restore the database from an ADBZ file, use the following command line for the `AlfaAdministrator-Console.exe`:

```
AlfaAdministratorConsole.exe-msalias <alias name>-DbUser <database user
name>-alfaLoginPassword <database user password>-db_restore-ArchiveInputFile
<file name and location>
```

The table below displays the command line options:

Command Line Option	Mandatory/ Default	Explanation
`-db_restore`	Mandatory	The command line must include the following command to archive the database to an ADBZ file: `-db_restore`
`-msalias <alias name>`	Mandatory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.

Command Line Option	Mandatory/ Default	Explanation
`-msaliasesfile` <Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-DbUser` <database user name>	Mandatory	Database user name for access to the Alfabet database on the database server.
`-DbPassword` <database user password>	Optional	Database user password for access to the Alfabet database on the database server. ⓘ If the access mode to the Alfabet database is Windows Authentication, the command line option `-DbUser` must be defined without a value for the user name and `-DbPassword` is not required.
`-ArchiveInput-File` <file name and location>	Mandatory	Specify the path to and name of the ADBZ file that contains the database content that shall be used to overwrite the current database.
`-IncludeAudit` <false|true>	Optional	To include the audit history tables in the ADBZ file into the database, set this parameter to `true`.
`-SqueezeAudit` <false|true>	Optional	If set to `true`, the history tables in the ADBZ file will be are scanned for obsolete entries. Any entries that were generated during, for example, the batch update of data via batch utilities without documenting any audit relevant changes are deleted from the audit tables restored from the ADBZ file.

Triggering Database Accessibility Monitoring Events

A mechanism is available that checks whether reading data from the Alfabet database and writing data to the Alfabet database is currently possible without causing any issues. The test is executed via a monitoring event that creates a test object, reads data about the test object, updates it, and deletes it. If errors occur during one of the processes, these will be logged.

The monitoring event can either be run once or scheduled to be executed in regular intervals of ten minutes.

Prior to exection of monitoring events, the functionality must be configured as described in the section Checking the Accessibility of the Alfabet Database.

After configuration, the execution of monitoring events can be triggered and stopped via the command line tool `AlfaAdministratorConsole.exe`.

To trigger a one-time execution of a monitoring event, run `AlfaAdministratorConsole.exe` with the following command line:

```
AlfaAdministratorConsole.exe-trigger_monitoring_event-msalias <alias name>-
DbUser <database user name> [-DbPassword <database user password>-timeout
<seconds>]
```

To start the recurring execution of monitoring events, run `AlfaAdministratorConsole.exe` with the following command line:

```
AlfaAdministratorConsole.exe-trigger_monitoring_event_chain-msalias <alias
name>-DbUser <database user name> [-DbPassword <database user password>]
```

To stop the recurring execution of monitoring events, run `AlfaAdministratorConsole.exe` with the following command line:

```
AlfaAdministratorConsole.exe-stop_monitoring_event_chain-msalias <alias
name>-DbUser <database user name> [-DbPassword <database user password>]
```

The table below displays the command line options:

Command Line Option	Manda-tory/ Default	Explanation
-<action type>	Manda-tory	Set the action type to one of the following: • `-trigger_monitoring_event` for the one-time execution of a monitoring event • `-trigger_monitoring_event_chain` for the recurring execution of monitoring events • `-stop_monitoring_event_chain` to stop the recurring execution of monitoring events.
`-msalias <alias name>`	Manda-tory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.
`-msalias-esfile <Alfabet con-figuration file path>`	Op-tional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-DbUser <data-base user name>`	Manda-tory	Specify the user name for access to the Alfabet database on the database server. For Oracle® databases, the user name is identical to the schema name.

Command Line Option	Manda- tory/ Default	Explanation
-DbPassword <database user password>	Manda- tory	Specify the password for access to the Alfabet database on the database server.

Triggering User Password Regeneration

For existing Alfabet users logging in via standard login, you can initially set or reset a password via the command line without knowledge about the current or the new password.

After having run the command line, two emails will be sent to each user defined in the command line. An email informing the user about the user name specified for login and a link to the login screen for first login. A second email containing an automatically generated password. Immediately after having entered the user name and password in the login screen for first login, the user is prompted to change his/her pass- word.

> (i) For details about the configuration of standard login and the assignment of user passwords for standard login, see Configuring Standard Login.

For existing Alfabet users, the password can be regenerated with one of the following command lines.

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <password>-db_regeneratepassword-userNames
<Alfabet user name,Alfabet user name>
```

Or:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <password>-db_regeneratepassword-userRefs
<Alfabet user REFSTR,Alfabet user REFSTR>
```

The table below displays the command line options:

Command Line Op- tion	Man- da- tory/ De- fault	Explanation
-db_regenerate- password	Man- datory	To regenerate the password of one or multiple users, the tool must be started with the option: `-db_regeneratepassword`

Command Line Option	Mandatory/Default	Explanation
`-msalias` <alias name>	Mandatory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.
`-msaliasesfile` <Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName` <Alfabet user name>	Mandatory	Alfabet user name for login. ⓘ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPassword` <password>	Optional	Password of the Alfabet user defined with `-alfaLoginName`.
`-userNames` <Alfabet user name>	Optional	Specify one Alfabet user name or multiple Alfabet user names in a comma-separated list to regenerate the password of all specified users. The users can be identified either by using this command line parameter or by using the `-userRefs` command line parameter.
`-userRefs` <Alfabet user `REFSTR`>	Optional	Specify the `REFSTR` of one Alfabet user the `REFSTR` of multiple Alfabet users in a comma-separated list to regenerate the password of all specified users. Users are stored in the object class `Person`. The users can be identified either by using this command line parameter or by using the `-userNames` command line parameter.

Resetting a User Password

For an existing Alfabet user logging in via standard login, a user password can be initially assigned or changed via the command line. If the user already has a password assigned, the administrator has to know the password to run the command line.

After having run the command line, the user administrator will know the user password. It is recommended that you select the **Change Password** checkbox in the **User** editor in the **User Management** functionality of the connected server alias for the user to force the user to change the password set via the command line after first login.

> ⓘ For details about the configuration of standard login and the assignment of user passwords for standard login, see [Configuring Standard Login](#).

For an existing Alfabet user, the password for standard login can be set or changed via the following command line:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <password>-db_setuserpassword-UserName <Alfabet
user name>-CurrentPassword <current user password>-NewPassword <new user
password>-RepeatNewPassword <new user password>
```

The table below displays the command line options:

Command Line Option	Manda-tory/ Default	Explanation
`-db_setuserpassword`	Manda-tory	To change the password for an existing Alfabet user: `-db_setuserpassword`
`-msalias <alias name>`	Manda-tory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.
`-msaliasesfile <Alfabet configuration file path>`	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName <Alfabet user name>`	Manda-tory	Alfabet user name for login. > ⓘ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPassword <password>`	Optional	Password of the Alfabet user defined with `-alfaLoginName`.
`-UserName <Alfabet user name>`	Manda-tory	Specify the user name for access to the Alfabet database on the database server. For Oracle® databases, the user name is identical to the schema name.
`-CurrentPassword <current user password>`	Manda-tory	Specify the current password of the Alfabet user. If the user do not have a password assigned, the parameter must be added to the command line without beeing followed by a user password.

Command Line Option	Manda- tory/ Default	Explanation
-NewPassword <new password>	Manda- tory	Specify the new password of the Alfabet user.
-RepeatNewPassword <new password>	Manda- tory	Specify the new password of the Alfabet user.

Anonymizing Data of Selected Users

This functionality anonymizes data for one or multiple selected users, which means for one or multiple se- lected object of the object class `Person`.

Data for a selected user is only anonymized if anonymization is enabled in the configuration of the object class `Person` and if the user is not explicitly excluded from anonymization.

⚠️ Anonymizing data is a sensible process that might disrupt database integrity. It cannot be re- verted! **Always back up the Alfabet database prior to triggering data anonymization!**

⚠️ If the anonymized user is the **Last Update User** or **Creator** of any configuration object subordi- nate to the **Classes** explorer node in the **Meta-Model** tab, the connections of all currently run- ning Alfabet components with the Alfabet database will be terminated and the database will be locked during the anonymization process. The Alfabet components need to be restarted after- wards.

To anonymize user data in an Alfabet database, use the following command line for the `AlfaAdministra- torConsole.exe`:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <user password>-db_anonymize_user-userRefs
<PersonREFSTR,PersonREFSTR>
```

The table below displays the command line options:

Command Line Op- tion	Manda- tory/ Default	Explanation
-db_anony- mize_user	Manda- tory	To anonymize user data for selected users in the Alfabet database, start the console application with 　　　　-db_anonymize_user

Command Line Option	Mandatory/ Default	Explanation
-userRefs <PersonREFSTR,Person-REFSTR>	Mandatory	Specify the value of the property REFSTR of the user that shall be anonymized. User data is stored in the object class Person. If the data of multiple users shall be anonymized, the REFSTR values can be listed comma-separated. The REFSTR property is not displayed on standard views in Alfabet. To see the REFSTR of a user, you can define a configured report in Alfabet Expand or read the information via RESTful service requests.
-msalias <alias name>	Mandatory	Enter the server alias name as specified in the AlfabetMS.xml configuration file for access to the database.
-msaliasesfile <Alfabet configuration file path>	Optional	If the AlfabetMS.xml configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the AlfabetMS.xml file must be specified with this parameter.
-alfaLoginName <Alfabet user name>	Mandatory	Alfabet user name for login. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=True) for the user.
-alfaLoginPassword <user password>	Optional	Alfabet login password.

Anonymizing Object Data

This functionality anonymizes all values for all object class properties configured to be anonymized in the Alfabet Meta-Model of the current Alfabet database.

⚠ Anonymizing data is a sensible process that might disrupt database integrity. It cannot be reverted! **Always back up the Alfabet database prior to triggering data anonymization!**

ⓘ For details about the configuration required to anonymize data and the consequences of anonymization, see *Anonymizing Data* in the reference manual *Configuring Alfabet with Alfabet Expand*.

To anonymize all relevant data in an Alfabet database, use the following command line for the AlfaAdministratorConsole.exe:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <user password>-db_anonymize_mm
```

The table below displays the command line options:

Command Line Option	Manda-tory/ Default	Explanation
`-db_anonymize_mm`	Manda-tory	To anonymize all data configured to be anonymized in an Alfabet database, start the console application with `-db_anonymize_mm`
`-msalias <alias name>`	Manda-tory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.
`-msaliasesfile <Alfabet configuration file path>`	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName <Alfabet user name>`	Manda-tory	Alfabet user name for login. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPass-word <user password>`	Optional	Alfabet login password.

Releasing a Database Restricted Mode

During critical operations such as updating or restoring the Alfabet database, a restricted mode is set for the connection to the Alfabet database that inhibits users other than the current user to access the Alfabet database.

The restricted mode automatically ends with the end of the critical operation. However, it is possible that the restricted mode may persist even if the critical operation has ended. In this case, you will need to release the restricted mode for the relevant database using the Alfabet Administrator.

To release the restricted mode, use the following command line for the `AlfaAdministratorConsole.exe`:

```
AlfaAdministratorConsole.exe-msalias <alias name>-DbUser <username>-
DbPassword <password>-db_releaserestrictedmode
```

The table below displays the command line options:

Command Line Option	Mandatory/ Default	Explanation
`-db_re-leaserestricted-mode`	Manda-tory	To release the restricted mode, start the console application with `-db_releaserestrictedmode`
`-msalias <alias name>`	Manda-tory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.
`-msaliasesfile <Alfabet configuration file path>`	Op-tional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-DbUser <user name>`	Manda-tory	User name for access to the Alfabet database on the database server. ⓘ If the access mode to the Alfabet database (27) is Windows Authentication, the command line option `-DbUser` must be defined without a value for the user name and `-DbPassword` is not required.
`-DbPassword <user password>`	Op-tional	Password for access to the Alfabet database on the database server.

Rebuilding Indexes on Database Tables

ADIF imports and rescan of indicators via the command line tool `RescanIndicatorsConsole.exe` or via the **Job Scheduler** functionality may trigger a high number of single database insert and delete transactions. This can have a negative impact on fragmentation of indices. As a result, CPU usage is increased and performance is reduced. It is recommended to rebuilt indices in regular intervals for object classes subject to ADIF import or rescan of indicators.

To rebuild indices for a number of object classes defined by object class name, use the following command line for the `AlfaAdministratorConsole.exe`:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <user password>-rebuild_indices-class_names
<comma separated list of object class names>
```

To rebuild indices for a number of database tables identified by table name, use the following command line for the `AlfaAdministratorConsole.exe`:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <user password>-rebuild_indices-table_names
<comma separated list of database table names>
```

To rebuild indices for all database tables, use the following command line for the `AlfaAdministrator-Console.exe`

The table below displays the command line options:

Command Line Option	Mandatory/ Default	Explanation
`-rebuild_class-indices`	Mandatory	To rebuild indices for database tables in the Alfabet database, start the console application with `-rebuild_indices`
`-class_names` <comma separated list of object class names>	It is mandatory to specify either-`class_names`,`-table_names`, or-`all`	Define the object classes for that an index shall be rebuild as a comma separated list of object class names. For example: `-class_names ApplicationGroup,OrgaUnit`
`-table_names` <comma separated list of database table names	It is mandatory to specify either-`class_names`,`-table_names`, or-`all`	Define the database tables for that an index shall be rebuild as a comma separated list of database table names. For example: `-table_names APPLICATIONGROUP, RELATIONS`
`-all` <TRUE/FALSE>	It is mandatory to specify either-`class_names`,`-table_names`, or-`all`	Add this command line parameter and set it to `TRUE` to rebuild all indices.
`-msalias` <alias name>	Mandatory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database.
`-msaliasesfile` <Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName` <Alfabet user name>	Mandatory	Alfabet user name of a named Alfabet user for login. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.

Command Line Option	Mandatory/ Default	Explanation
`-alfaLoginPass-word` <user password>	Optional	Alfabet login password.

Updating Maintenance Window Definitions for the Job Schedule Functionality

Alfabet offers a **Job Schedule** functionality to schedule the regular execution of ADIF import, ADIF export, and batch executables that are part of the Alfabet components. Maintenance work on the Alfabet components, like for example update of the meta-model, conflicts with the execution of scheduled jobs. To ensure that no scheduled jobs are executed during maintenance, maintenance windows can be defined in Alfabet. Job schedules can then be configured to read the current maintenance window specification and schedule any job that is due during a maintenance window to start 1 minute after the end time of the maintenance window instead.

Maintenance windows are defined in an XML with a defined structure. The definition can either be configured directly in the XML object **MaintenanceWindows** that is part of the meta-model configuration object configurable in Alfabet Expand or the Alfabet Administrator, or it can be defined in an XML file located on the local file system and uploaded to the meta-model configuration in the Alfabet database with the tool AlfabetAdministratorConsole.exe.

For information about how to define the XML content of the file containing the maintenance window definitions, see Defining Maintenance Windows for Scheduled Jobs.

To add the content of your external file to the already available maintenance window definition in the Alfabet database:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <user password>-add_maintenance_window
<file>.xml
```

To overwrite the already available maintenance window definition in the Alfabet database with the content in your external file:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <user password>-replace_maintenance_windows
<file>.xml
```

The table below displays the command line options:

Command Line Option	Mandatory/ Default	Explanation
`-<action type> <file>.xml`	Mandatory	Enter the action type to be performed followed by the absolute path to the file that contains the data. Allowed action types are: • `-add_maintenance_window` to add the content in the external XML file to the maintenance window definition in the Alfabet database. • `-replace_maintenance_window` to replace the maintenance window definition in the Alfabet database with the content in the external XML file.
`-msalias <alias name>`	Mandatory	Enter a server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database. Please note that the process involves execution of an ADIF import in the background. If you connect to the database with a server alias that is configured to use an Alfabet Server for the execution of ADIF jobs, this Alfabet Server must be running to execute the update.
`-msaliasesfile <Alfabet configuration file path>`	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName <Alfabet user name>`	Mandatory	Alfabet user name for login. ⚠️ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPassword <user password>`	Optional	Alfabet login password.

Defining Maintenance Windows for Scheduled Jobs

Alfabet offers a **Job Schedule** functionality to schedule the regular execution of ADIF import, ADIF export, and batch executables that are part of the Alfabet components. Maintenance work on the Alfabet components, like for example update of the meta-model, conflicts with the execution of scheduled jobs. To ensure that no scheduled jobs are executed during maintenance, maintenance windows can be defined in Alfabet. Job schedules can then be configured to read the current maintenance window specification and schedule any job that is due during a maintenance window to start 1 minute after the end time of the maintenance window instead.

Maintenance windows can either be configured to be recurring once a week or once a month or they can be scheduled for a specified date. All specifications refer to the time of the server host.

There are two ways for system administrators to define maintenance windows:

- Maintenance windows can be configured in the XML object **MaintenanceWindows** in the Alfabet Administrator.

  ⓘ XML objects can only be edited in the Alfabet Administrator if the attribute **Visible in Administrator** of the XML object is set to `True` in Alfabet Expand. If you do not see the XML object in the Alfabet Administrator, the attribute is set to `False`. Ask a solution designer with access permissions to Alfabet Expand to change the attribute setting.

- Maintenance windows can be defined in an XML file located on the local file system and uploaded to the XML object **MaintenanceWindows** via the command line tool `AlfaAdministratorConsole.exe`.

The following information is available:

- [Defining Maintenance Windows in Alfabet Administrator](#)

- [Importing Maintenance Window Definitions from an External XML File](#)

Defining Maintenance Windows in Alfabet Administrator

Maintenance windows can be configured in the XML object **MaintenanceWindows** in the Alfabet Administrator:

1) In the explorer of the Alfabet Administrator, right click the server alias of the Alfabet Web Application and select **Connect**.

2) Enter your login credentials for login to the Alfabet database on the database server.

3) Click **XML Objects** in the expanded server alias node.

4) in the table on the right, double-click **MaintenanceWindows**. The XML editor opens with an empty root XML element:

   ```
   <MaintenanceWindows></MaintenanceWindows>
   ```

5) For each maintenance window that you want to define, add a child XML element `MaintenanceWindow` to the XML element `MaintenanceWindows`.

6) The definition of the XML attributes for the XML element `MaintenanceWindow` depends on the scheduling options:

To define a recurring maintenance window, add the following XML attributes:

- `DayOfWeek`: If the maintenance window occurs once a week, define the day of the week that the maintenance window shall occur. Allowed values are the English names of the days of the week starting with a capital letter.

- `DayOfMonth`: If the maintenance window occurs once a month, define the day of the month that the maintenance window shall occur as integer.

- `StartTime`: Define the start time of the maintenance window in the twenty-four hour time in the format hh:mm.

- `EndTime`: Define the end time of the maintenance window in the twenty-four hour time in the format hh:mm.

Example for maintenance windows recurring once a week and once a month:

```
<MaintenanceWindows>

    <MaintenanceWindow DayOfWeek="Friday" StartTime="20:30"
    EndTime="23:00"/>

    <MaintenanceWindow DayOfMonth="12" StartTime="08:00"
    EndTime="12:00"/>

</MaintenanceWindows>
```

To define a maintenance window for a defined date, add the following XML attributes:

- `Format`: Define the date format that you will use for definition of the date for the maintenance window. the XML attribute is optional. If it is not defined, the date format defined in the culture settings that are configured to be the primary culture are used.Allowed formats are:

  - MM/dd/yyyy

  - dd/MM/yyyy

  - M/d/yyyy

  - d/M/yyyy

- `Date`: Define the date the maintenance window will occur in the format defined with the XML attribute `Format`.

- `StartTime`: Define the start time of the maintenance window in the twenty-four hour time in the format hh:mm.

- `EndTime`: Define the start time of the maintenance window in the twenty-four hour time in the format hh:mm.

Example for a one time maintenance window:

```
<MaintenanceWindows>

    <MaintenanceWindow Date="12/13/2019" Format="MM/dd/yyyy"
    StartTime="08:00" EndTime="12:00"/>

</MaintenanceWindows>
```

7) Click **OK**.

Importing Maintenance Window Definitions from an External XML File

Maintenance windows can be defined in an XML file on the local file system and imported into the Alfabet database via the tool AlfaAdministratorConsole.exe.

There are two methods for maintaining the data:

- The complete specification of all maintenance windows can be configured in the file and the data in the Alfabet database can be overwritten with this content after a change has been performed.

- Single maintenance windows can be defined in separate files and the information can be added to the maintenance window definitions already uploaded to the Alfabet database. Please note that this is not a merging process. If you define a maintenance window that is already defined in the maintenance window definitions in the Alfabet database, the maintenance window definition will be doubled.

To define the maintenance windows in an external XML file, create an XML file with any text editor and store it on the local file system. Write the following into the file:

1) The root XML element must be `MaintenanceWindows`:

   ```
   <MaintenanceWindows></MaintenanceWindows>
   ```

2) For each maintenance window that you want to define, add a child XML element `MaintenanceWindow` to the XML element `MaintenanceWindows`.

3) The definition of the XML attributes for the XML element `MaintenanceWindow` depends on the scheduling options:

   To define a recurring maintenance window, add the following XML attributes:

   - `DayOfWeek`: If the maintenance window occurs once a week, define the day of the week that the maintenance window shall occur. Allowed values are the English names of the days of the week starting with a capital letter.

   - `DayOfMonth`: If the maintenance window occurs once a month, define the day of the month that the maintenance window shall occur as integer.

   - `StartTime`: Define the start time of the maintenance window in the twenty-four hour time in the format hh:mm.

   - `EndTime`: Define the end time of the maintenance window in the twenty-four hour time in the format hh:mm.

     Example for maintenance windows recurring once a week and once a month:

     ```
     <MaintenanceWindows>

         <MaintenanceWindow DayOfWeek="Friday" StartTime="20:30"
         EndTime="23:00"/>

         <MaintenanceWindow DayOfMonth="12" StartTime="08:00"
         EndTime="12:00"/>

     </MaintenanceWindows>
     ```

   To define a maintenance window for a defined date, add the following XML attributes:

- `Format`: Define the date format that you will use for definition of the date for the maintenance window. the XML attribute is optional. If it is not defined, the date format defined in the culture settings that are configured to be the primary culture are used.Allowed formats are:

  - MM/dd/yyyy

  - dd/MM/yyyy

  - M/d/yyyy

  - d/M/yyyy

- `Date`: Define the date the maintenance window will occur in the format defined with the XML attribute `Format`.

- `StartTime`: Define the start time of the maintenance window in the twenty-four hour time in the format hh:mm.

- `EndTime`: Define the start time of the maintenance window in the twenty-four hour time in the format hh:mm.

Example for a one time maintenance window:

```
<MaintenanceWindows>

    <MaintenanceWindow Date="12/13/2019" Format="MM/dd/yyyy"
    StartTime="08:00" EndTime="12:00"/>

</MaintenanceWindows>
```

To import the data into the Alfabet database, use the tool `AlfabetAdministratorConsole.exe`:

Executable	`AlfaAdministratorConsole.exe` located in the ***Programs*** subdirectory of the Alfabet installation directory
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see <u>Standard Logging for Alfabet Batch Utilities</u>.
Command line help	Start executable with `-h` or `-help`

To add the content of your external file to the already available maintenance window definition in the Alfabet database:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <user password>-add_maintenance_window
<file>.xml
```

To overwrite the already available maintenance window definition in the Alfabet database with the content in your external file:

```
AlfaAdministratorConsole.exe-msalias <alias name>-alfaLoginName <Alfabet
user name>-alfaLoginPassword <user password>-replace_maintenance_windows
<file>.xml
```

The table below displays the command line options:

Command Line Option	Mandatory/Default	Explanation
`-<action type> <file>.xml`	Mandatory	Enter the action type to be performed followed by the absolute path to the file that contains the data. Allowed action types are: • `-add_maintenance_window` to add the content in the external XML file to the maintenance window definition in the Alfabet database. • `-replace_maintenance_window` to replace the maintenance window definition in the Alfabet database with the content in the external XML file.
`-msalias <alias name>`	Mandatory	Enter a server alias name as specified in the `AlfabetMS.xml` configuration file for access to the database. Please note that the process involves execution of an ADIF import in the background. If you connect to the database with a server alias that is configured to use an Alfabet Server for the execution of ADIF jobs, this Alfabet Server must be running to execute the update.
`-msaliasesfile <Alfabet configuration file path>`	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName <Alfabet user name>`	Mandatory	Alfabet user name for login. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPassword <user password>`	Optional	Alfabet login password.

Chapter 8: Data Exchange with Third-Party Applications

Companies may use other databases that contain information similar to the data in the Alfabet database or use applications that require input that is already available in the Alfabet database. Therefore Alfabet developed interfaces that allow access to the data in the Alfabet database for use in external applications.

In the following, an overview of the interfaces, structured by functionality, is given, to enable you to select the best solution matching your demands. The interfaces offer the following functionality:

Providing Current Data from the Alfabet database to External Applications

The Alfabet database gives a full overview over the IT infrastructure. This might be a useful source of information for external applications that rely on up-to-date data.

Depending on the external application and the kind of input required, there are three different ways to get up-to-date information from the Alfabet database.

> It is also possible to publish data directly from the Alfabet user interface. For more information, see the section *Exporting Data* in the reference manual *Getting Started with Alfabet*.

- **Batch export of data to XML, HTML, CSV, PDF files, Microsoft® Word files, or Microsoft® Excel® files** All other utilities search the Alfabet database for relevant data by means of Alfabet queries defined in the Alfabet query language 500 or native SQL queries. Depending on the tool you are working with, the queries are either defined in an XML configuration file for the batch utility or in either export definitions or configured reports defined in the configuration tool Alfabet Expand.

  A number of batch utilities are available to export data from the Alfabet database to files of formats that can either be used as input for third-party applications or for the publication of data, for example, in the company's intranet. The batch jobs can be executed in regular intervals to ensure that up-to-date data is available for external use.

  The Alfabet Publication Framework (APF) fills bookmarks within customer defined a Microsoft Word template file with data from the Alfabet database and reports from the Alfabet user interface according to a customer defined mapping scheme. The resulting document is then stored either as a Microsoft Word file or a PDF file.

  The Alfabet Data Integration Framework (ADIF) can be used for both import and export of data to and from different file formats. The export is based on an export definition that is customer configured using native SQL queries.

  The following table provides an overview of the available tools and interfaces for batch data export:

(Interface/)Executable	Exported Data	Data Export to	Export Based on	For more information see...
Alfabet Publication Framework (APF)/Publication-Console.exe	Information about data in the Alfabet database and Alfabet reports	Microsoft Word or PDF	Customer defined Microsoft Word templates and mapping	For information about the configuration of Microsoft Word templates and mapping of data to the template, see

(Interface/)Executable	Exported Data	Data Export to	Export Based on	For more information see...
	and views integrated into a word template that allows to add design and text to the data.		defined in Alfabet Expand to map bookmarks in the template with data in Alfabet.	*Publishing Data In Microsoft® Word Format* in the reference manual *Configuring Alfabet with Alfabet Expand*. For information about the execution of export via a batch job see [Exporting Data to Microsoft® Word or PDF Format via the Alfabet Publication Framework](#).
Alfabet Data Integration Framework (ADIF)/ADIF_Console.exe	Information about data in the Alfabet database in either a tabular structure or as a customer defined XML.	XML, CSV, Microsoft Excel	Native SQL query-based ADIF export scheme defined in Alfabet Expand.	For information about the configuration of data export via ADIF, see the reference manual *Alfabet Data Integration Framework*. For information about the execution of export via a batch job, see [Starting the Publication Console Application](#).
QueryExecutor.exe	Information about data in the Alfabet database in a tabular structure.	XML, HTML, or Microsoft Excel	An Alfabet query-based XML export definition.	[Exporting Data to XML, HTML, or Microsoft® Excel® Format with the QueryExecutor.exe](#)
AlfaConsoleImportExport.exe	Information about data in the Alfabet database in an XML following an Alfabet defined schema.	XML	An export definition defined in Alfabet Expand.	[Exporting Data to XML Format with AlfaConsoleImportExport.exe](#)
AlfaExportUtil.exe	Information about data in the Alfabet database in a tabular structure.	CSV	An XML export definition that can contain WHERE statements from query language.	[Exporting Data in Value-Separated Format (CSV)](#)

- **Batch export of data to an external database**

    The Alfabet Data Integration Framework enables batch export of Alfabet data to an external database. The export is based on an export scheme defined in Alfabet Expand. In the export scheme, the customer can define the restructuring of the Alfabet data to meet the structure and import conditions of the external database. The export is executed via the console application ADIF_Console.exe.

    For information about the configuration of data export via ADIF, see the reference manual *Alfabet Data Integration Framework*.

    For information about the execution of export via a batch job see <u>Importing or Exporting Data from External Database Tables or Files via ADIF</u>.

- **Interfaces for external applications to search the Alfabet database for current data**

    External applications can perform run-time evaluation of current data by searching the Alfabet database either using SQL query language or SOAP based access to the Alfabet database.

Provided Interface	Database Access	Mechanisms used	For more information see...
Alfabet Web Services	SOAP based access via the Alfabet Web Services connecting to the Alfabet Server.	SOAP/WSDL	For information about the Alfabet Web Services, see the reference manual *Web Services for Alfabet*.
none	The external application can directly access the Alfabet database via the database server and request data via SQL query language.	SQL	For information, see <u>Accessing the Alfabet Database with External Applications</u>.

One-Time Import of Data from External Data Sources to the Alfabet Database

When implementing Alfabet, the information that must be stored in the Alfabet database is normally already available in other data sources (for example, in databases or service registries). Entering data manually to the new Alfabet database is time consuming and ineffective. Therefore Software AG provides mechanisms to batch import data from external sources to the Alfabet database. The same mechanisms can also be used to integrate data from a new database to the already existent Alfabet database (for example, in case of a company merger(.

> ⚠ It is not possible to import data to a Alfabet database by means of tools and mechanisms not provided by Alfabet. Data integrity cannot be maintained without the data integrity checks included in the import interfaces provided by Software AG.

The following data import mechanisms are available:

Interface	External Data Source	Integration mechanisms:	For more information see...
External Data Integration Interface	LDAP, Microsoft SQL Server, Oracle	Data import via the console application `ExternalSourcesSynchronization.exe` according to an XML configuration that maps data between the external data source and the Alfabet database.	[Integrating Data from External Sources](#)
Alfabet Data Integration Framework (ADIF)	LDAP, Microsoft SQL Server, Oracle, Microsoft Access, PostGreSQL	Data import via the console application ADIF_Console.exe according to an import scheme defining import via SQL queries. In the ADIF import scheme, data can be processed prior to import definition to fit the requirements of the Alfabet database and the customer performing the import.	For information about the configuration of data import via ADIF, see the reference manual *Alfabet Data Integration Framework*. For information about the execution of import via a batch job, see [Importing or Exporting Data from External Database Tables or Files via ADIF](#).

Run-Time Synchronization with an External Data Source

In some cases, data that is relevant for Alfabet must be maintained in external databases rather than in Alfabet because it is mainly used by other applications. For example, this could be relevant for the following contexts:

- Data about Alfabet users required in the Alfabet database to define access permissions are identical to a subset of data from the company's employee database. The employee data is maintained in the external employee database and the Alfabet database will be synchronized with the employee database at runtime to ensure that the user data in Alfabet is up-to-date.

- Service registries used to maintain a SOAP based architecture are available in the company and consistency between the data in the service registry and the data stored in the Alfabet database must be ensured.

Software AG provides interfaces that allow to synchronize data in the Alfabet database at runtime with the data of external data sources:

Interface	External Data Source	Integration mechanisms:	For more information see...
External Data Integration Interface	LDAP, Microsoft SQL Server, Oracle	Selectors in the Alfabet user interface can be configured to display data from external sources and data in the Alfabet database is updated on selection of an object.	[Integrating Data from External Sources](#)

Interface	External Data Source	Integration mechanisms:	For more information see...
		Authentication and authorization can be managed via the external data source.	

Making Alfabet Reports Available in the Intranet or via Emails

Alfabet reports provide useful information that a company might want to publish on the intranet to make it available for use outside Alfabet. Software AG provides various mechanisms that ease the publication of data outside Alfabet:

- **Publishing data in formats that can be used for publication in the intranet.**

  Data can be published by the Alfabet user directly from the Alfabet user interface:

Mechanism used to publish data	Published Data	Data Export To	Form more information see...
Alfabet Publication Framework (APF) available via a configured report or button in the Alfabet user interface.	Information about data in the Alfabet database and Alfabet reports and views integrated into a word template that allows to add design and text to the data.	Microsoft Word or PDF	For information about the configuration of Microsoft Word templates, mapping of data to the template and configuration of the user interface, see *Publishing Data In Microsoft® Word Format* in the reference manual *Configuring Alfabet with Alfabet Expand*.
Publish button in the toolbar of Alfabet views and object profiles.	Current view	HTML, PDF, Microsoft Excel file, Microsoft Word file, or, in case of graphics, an image file format.	*Exporting Data* in the reference manual *Getting Started with Alfabet*.

  Batch utilities also allow to publish reports about Alfabet data in file formats that can be used for publishing:

(Interface/)Executable	Exported Data	Data Export to	Export Based on	For more information see...
Alfabet Publication Framework (APF)/PublicationConsole.exe	Information about data in the Alfabet database and Alfabet reports	Microsoft Word or PDF	Customer defined Microsoft Word templates and mapping	For information about the configuration of Microsoft Word templates and mapping of data to the template, see

(Interface/)Executable	Exported Data	Data Export to	Export Based on	For more information see...
	and views integrated into a word template that allows to add design and text to the data.		defined in Alfabet Expand to map bookmarks in the template with data in Alfabet.	*Publishing Data In Microsoft® Word Format* in the reference manual *Configuring Alfabet with Alfabet Expand*. For information about the execution of export via a batch job see Exporting Data to Microsoft® Word or PDF Format via the Alfabet Publication Framework.
Alfabet Data Integration Framework (ADIF)/ADIF_Console.exe	Information about data in the Alfabet database in either a tabular structure or as a customer defined XML.	XML, CSV, Microsoft Excel	Native SQL query based ADIF export scheme defined in Alfabet Expand.	For information about the configuration of data export via ADIF, see the reference manual *Alfabet Data Integration Framework*. For information about the execution of export via a batch job, see Starting the Publication Console Application.
QueryExecutor.exe	Information about data in the Alfabet database in a tabular structure.	XML, HTML, or Microsoft Excel	An Alfabet query based XML export definition.	Exporting Data to XML, HTML, or Microsoft® Excel® Format with the QueryExecutor.exe
AlfaConsoleImportExport.exe	Information about data in the Alfabet database in an XML following an Alfabet defined schema.	XML	An export definition defined in Alfabet Expand.	Exporting Data to XML Format with AlfaConsoleImportExport.exe
AlfaExportUtil.exe	Information about data in the Alfabet database in a tabular structure.	CSV	An XML export definition that can contain WHERE statements from query language.	Exporting Data in Value-Separated Format (CSV)

- **Providing access to single views to the Alfabet interface from external web applications or emails.**

  A special link syntax is provided by Software AG that allows object profiles, page views or configured report to be opened from external web sites or from emails. For more information, see Links to Alfabet Views from External Applications.

  Additionally, Alfabet users can send a link to a Alfabet object profile or page view via email to any recipient using a functionality in the Alfabet user interface. (For more information, see the section *Sending and Receiving Express Views* in the reference manual *Getting Started with Alfabet*.)

Integrating External Reporting

Software AG provides a wealth of ready-to-use reports about the data in the Alfabet interface. Nevertheless, customers might require additional reports or reports in formats that are not supported by Alfabet or that take over data from multiple databases including Alfabet. Therefore, Software AG provides interfaces that enable the use of external reporting tools to generate reports about Alfabet data:

- **Direct SQL-based access to the Alfabet database** allows user to search for information and present it in an external report that is generated by means of a script or reporting tool. This enables companies to create any report about Alfabet data independent of the standard Alfabet reports. For more information, see Accessing the Alfabet Database with External Applications.

- **Links to Alfabet views from external reports** allows users to click a link in the external report in order to open a Alfabet view within Alfabet that is required in the context of the report. For more information, see Links to Alfabet Views from External Applications.

- **Dynamic Web links to external reports from within the Alfabet interface** allows users to view customized reports in the Alfabet interface. The reports are independent of Alfabet and can also include reports based on other databases (for example, an up-to-date list of customer contact data from an LDAP table). For more information, see Integration of External Reports into Alfabet .

The following figure gives an overview of the interfaces for data import, export and synchronization.

HTML   PDF   Microsoft ® Excel   Microsoft ® Word   Graphic (PNG, EMF, JPG, BMP, SVG)

Web Applications

links to the planningIT interface can be specified

alfabet Export Functionalities

alfabet Interface

Service Registry

links to the planningIT interface can be specified and external reports can be integrated into the planningIT interface

alfabet Web Application

alfabet Server/ Client

alfabet Interface for External Data Integration

External Reporting Server

SQL queries

External Database (LDAP, Oracle, Microsoft ® SQL Server)

alfabet Interface for External Data Integration and Batch Job: ExternalSourceSynchorinzation. exe

External Database (PostGreSQL, Microsoft ® Access)

CSV

Batch Job: AlfaExportUtil. exe

Microsoft ® Word

alfabet Publication Framework (aPF)

alfabet Data Integration Framework (aDIF)

CSV

PDF

Microsoft ® Excel

planningIT Database

XML

XML

Batch Job: AlfaConsole ImportExport. exe

Microsoft ® Excel ®

Batch Job: QueryExecutor.exe

alfabet Web Services

SOAP

External Application

HTML

Legend:

- alfabet components and internal communication between components
- external applications and external mechanisms
- external data formats/generated files
- tool/interface provided by alfabet for one-time import or export of data
- tool/interface provided by alfabet for runtime import or export of data

FIGURE: Overview of interfaces to external applications and data formats

For a detailed description of the individual interfaces, see:

- Importing or Exporting Data from External Database Tables or Files via ADIF

  - Providing Relevant Data and Configuration

  - Starting the ADIF Console Application

- Exporting Data to Microsoft® Word or PDF Format via the Alfabet Publication Framework

  - Triggering Publications via a Batch Utility

    - Providing Relevant Data and Configuration

    - Starting the Publication Console Application

Importing or Exporting Data from External Database Tables or Files via ADIF

The Alfabet Data Integration Framework (referred to as ADIF) is a technology to batch import, export, and manipulate very large amounts of data in the Alfabet database. Its interface is highly configurable by means of native SQL commands in combination with conversion procedures delivered by Software AG. Via ADIF, data can be imported from and exported to the following sources:

- external database tables

- CSV files

- XLS files

- XLSX files

- XML files

The ADIF interface allows data to be imported from any customer-defined format within the data source. Data can be integrated from multiple sources in one step. This means that you can, for example, configure the interface to import data that is partially derived from a database table and partially stored in XML files.

The way data is imported or exported is configured in XML based ADIF schemes. The schemes are configured via an interface provided by Alfabet in the tool Alfabet Expand.

> The ADIF functionalities require purchase of a special license. The interface for ADIF configuration is only available in Alfabet Expand if licensed by the customer.

The data can be manipulated during import or export via customer defined SQL commands. Database integrity in the Alfabet database is ensured by mechanisms implemented by Alfabet. These mechanisms also ease the configuration of the scheme.

The configuration of ADIF schemes is highly flexible and offers a wide range of functionalities. This is described in the reference manual *Alfabet Data Integration Framework*. The information below is limited to the documentation of the execution of import or export based on existing configured ADIF schemes.

After you have configured the ADIF interface and controlled your configuration via the ADIF debugger, you can use the configured ADIF scheme for data import or export or data manipulation either on request or on a regular basis.

Software AG provides a Windows® command line tool `ADIF_Console.exe` to execute import, export or manipulation of data.

> For security reasons, the following recommendations should be considered for the import of data to the Alfabet database or for export of data to external databases:
>
> - Test the import of data in a test environment prior to the import to a production environment for each import performed via the ADIF console application to ensure that the correct changes are applied.
>
> - Backup the production database prior to the import of data.

Executable	`ADIF_Console.exe` located in the ***Programs*** sub-directory of the installation directory of the Alfabet components.
Preconditions	An `AlfabetMS.xml` configuration file with a server alias configuration specifying the connection to the Alfabet database is available and the respective functionality is configured.
Logging	Log information is written to a log file and to the Alfabet database. Logging is configurable via the ADIF scheme used or in the command line. For more information about logging, see *Configuring Logging Parameters* and the list of command line options below.
Command line help	Start executable with `-h` or `-help`

Executables for batch jobs can be started in a command line or by means of a Windows® batch job.

When executing a Windows batch job, the execution time for a batch job is defined with the help of the Windows scheduler for batch jobs. When starting the executable with a command line, the batch job is executed immediately.

The setting in the **Application Server** tab must be identical in all server alias configurations of all Alfabet components including batch utilities.

Providing Relevant Data and Configuration

The ADIF console application requires access to the following files to be executed:

A fully configured ADIF scheme

The ADIF import or export scheme must be completely configured and debugged before starting the data import or export. The ADIF scheme can be provided as an XML file on the local host or as part of the Alfabet database.

> ⚠ When using an ADIF scheme from an XML file, you must ensure that no ADIF scheme with a **Name** attribute exists in the Alfabet database that is identical to the **Name** attribute of the ADIF scheme in the XML file.
>
> The ADIF scheme in the XML file is temporarily loaded to the Alfabet database during import and deleted from the Alfabet database after the import/export process. If an ADIF scheme with the same **Name** is available in the Alfabet database when starting the ADIF console application, this ADIF scheme will be overwritten by the ADIF scheme in the XML file and deleted after the import/export process.

An AlfabetMS.xml configuration file

For access to the Alfabet database, an `AlfabetMS.xml` configuration file with an alias configuration must be available.

The way batch jobs are executed depends on the setting of the attributes in the Application Server tab of the server alias configurations of the Alfabet components:

- If **Use Event Queue for All Jobs** is selected, all server processes are scheduled via event queueing. The ADIF console application schedules a job for execution of the ADIF job in the Alfabet database. The Alfabet Server scans the queue of jobs to be processed in regular intervals and processes the incoming jobs. More than one ADIF job can be processed in parallel via different threads. The Alfabet Server only starts jobs in parallel if they do not write data into the same database table in the Alfabet database.

  This is the recommended option that will be maintained in future releases.

- If **Use Application Server and Net Remoting Service** is selected, the ADIF console application hands the job over to the Alfabet Server for execution. The ADIF console application waits until the Alfabet Server finishes the execution and then retrieves the log and the success status from the Alfabet Server. The Alfabet Server executes only one ADIF job at the same time. By default, if you try to start an ADIF job while another job is running, your new job will be terminated with an error message. However, you can configure the ADIF console to queue jobs for execution. A maximum queue time must be defined in the command line to queue the job. If a queue time is defined, and a job is started while another job is still running, the new job will be queued and will start when the first job is finished. If the job is queued longer than the defined maximum queue time, it is deleted from the queue without being executed.

  If a remote alias is used for execution of the ADIF console application, the corresponding Alfabet Server must be running. If a server alias is used, all other Alfabet applications with access to the Alfabet database must be stopped.

  This method will be deprecated in the future because it will not be compatible with the upcoming.NET Core component.

When starting the ADIF console application, the alias configuration name must be specified in the command line. If the `AlfabetMS.xml` configuration file containing the alias configuration is not located in the working directory of the ADIF console application, the path to the file must additionally be specified in the command line.

For import: Access to data that shall be imported

If the data shall be imported from files, the files containing the data must be located either in a single *.zip file or in a single import directory. If files are located in an import directory, the ADIF console application will create a ZIP file containing all files located in the import directory for use in data import as part of the import process.

> ⚠ When import shall be performed from an XML file referring to a DTD file, the DTD must also be located in the import ZIP file or import directory respectively.

Files are uploaded to the Alfabet database and deleted from the database after import was performed. Whether the import files on the local host are changed by the import process depends on the import scheme configuration. For more information, see *Configuring Data Import with ADIF*.

If data is imported from multiple import files in one ADIF import and a specified import files are missing, import from all other files is performed and a warning is written to the log file that import of data from the missing file could not be performed.

Starting the ADIF Console Application

Start `ADIF_Console.exe` with the command line options listed in the table below to perform a data import based on an ADIF import scheme. If a command line option requires a parameter definition and the parameter includes whitespaces, the parameter must be written in inverted commas. For example:

```
ADIF_Console.exe-import-msalias Alfabet-scheme "Application Import"
```

For the import of data, you must start `ADIF_Console.exe` with at least:

- a specification of the alias configuration defining access to an Alfabet Server or a Alfabet database,

- the import scheme and, if data is imported from files,

- the location of the import files in the local file system:

> 👁
> ```
> ADIF_Console.exe-import-msalias <alias name> [-scheme <scheme name>-
> importfile <file name>.zip]
> ```

For export of data, you must start `ADIF_Console.exe` with at least:

- a specification of the alias configuration defining access to an Alfabet Server or a Alfabet database,

- the export scheme and, if data is exported to files,

- the location in the local file system where the export files shall be stored

```
ADIF_Console.exe-export-msalias <alias name>-alfaLoginName <user
name>-alfaLoginPassword <user password> [-scheme <scheme name>-
importfile <file name>.zip]
```

Command Line Option	Mandatory/ Default	Explanation

-<action>	Mandatory	To import data, start the console application with
		
		    `-import`
		
		To export data, start the console application with
		
		    `-export`
		
		In the command line help that can be called with the `-h` command line option, a `-update` action type is also listed next to `-import` and `-export`. This option is for internal use by Software AG only and must not be used.

-msalias <alias name>	Mandatory	Enter the alias name as specified in the `AlfabetMS.xml` configuration file.
-msaliasesfile < Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
-alfaLoginName <user name>	Mandatory	User name for access to the Alfabet database. The user specified by this login name will be written to the audit history of changed objects as responsible for changing the data. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
-alfaLoginPassword <user password>	Optional	Password for access to the Alfabet database.
-scheme	Optional	Specify the name of the ADIF scheme stored in the Alfabet database that contains the configuration of the data import/export. **NOTE**: It is mandatory to specify this option or the option `-schemefile`. Only one of the options can be specified in a command line. Which of the options is applicable will depend on the location of the ADIF scheme.

Command Line Option	Mandatory/ Default	Explanation
-schemefile	Optional	Specify the name of and path to the XML file of the ADIF scheme that contains the configuration of the data import/export. **NOTE**: It is mandatory to specify this option or the option-scheme. Only one of the options can be specified in a command line. Which of the options is applicable depends on the location of the ADIF import scheme. ⚠ If a scheme with the same name as the scheme in the external XML file exists in the Alfabet database when running the job, the scheme in the Alfabet database will be deleted. This mechanism is implemented to ensure that only one version of a scheme exists. If the scheme is called via an external XML file, it is supposed that the scheme with the same name in the Alfabet database is outdated and no longer used.
-waittime	Optional	Only applicable if Alfabet components are using event queueing for processing of ADIF jobs. Specify the time in minutes that the process shall wait if another ADIF process is running. Two processes cannot run in parallel. A new process is only queued for execution if the command line option-waittime is set. If the currently running process is not finished during the configured wait time, the queued process will be deleted. If-waittime is not set and a process is started while another process is currently running, the new process is terminated, and an error message is given.
-idletime	Optional	Only applicable if Alfabet components are using event queueing for processing of ADIF jobs. The batch utility waits for alive messages from the Alfabet Server and if none are received within 60 seconds, the job will be cancelled. The allowed wait time for alive messages can be changed by defining a new time interval in seconds with-idletime.
-synchronously	Optional	Only applicable if the Alfabet components are using event queueing for processing of ADIF jobs. If this parameter is added to the command line, the ADIF console application will check the event queue for the status of the scheduled event until the event is processed and will return the status of the finished ADIF job execution.
-<variable name>	Optional	When the ADIF scheme is configured to use variables, the variables can be specified in the command line. For each variable, a separate command line option must be specified as

Command Line Option	Mandatory/ Default	Explanation

		`-<variable name> <value>`
		
		For example:
		
		`-@fiscalyear 2009`
		
		For more information about the use of variables, see *Configuring Execution Dependent on Current Parameters* in the description of the export configuration or *Configuring Import Dependent on Parameters* in the description of the import configuration.

Import-specific command line options

Command Line Option	Mandatory/ Default	Explanation
`-importfile`	Optional	Specify the name of and path to the ZIP file containing the files from which data shall be imported. **NOTE**: If the import scheme configuration defines import from files, it is mandatory to specify this option or the option `-importdir`. Only one of the options can be specified in a command line.
`-idletime`	Optional	Specify the name of and path to the directory containing the files from which data shall be imported. **NOTE**: If the import scheme configuration defines import from files, it is mandatory to specify this option or the option `-importfile`. Only one of the options can be specified in a command line.

Export-specific command line options

Command Line Option	Mandatory/ Default	Explanation
`-exportfile`	Optional	Specify the name of and path to the ZIP file to which the data will be written. If the file does not exist, it will be created; if it does exist, it will be overwritten. If the path does not exist, the ADIF export cannot be executed. The export mechanisms can create files, but it cannot create directories. **NOTE**: If the ADIF export scheme configuration defines export to files, it is mandatory to specify this option.
`-unzip <true|false>`	Optional	If set to `true`, the ZIP file to which the export data is written is unzipped after the export. The files with the export data are then located in the directory specified as the location for the ZIP file as well as in the ZIP file. If set to `false`, the ZIP file to which the export data is written is not unzipped and the data is available in zipped format only.

Command Line Option	Mandatory/ Default	Explanation

Logging-specific command line options

Command Line Option	Mandatory/ Default	Explanation
`-logpath` <log file path>	Log file is stored in the working directory of the executable.	Specification of a path to a directory for storage of the log file.
`-logfile` <log file name>	<Executable>.log	Specification of the log file name. Allowed file extensions are LOG and TXT.
`-logverbose`		If `-logverbose` is set, additional information about the running process is logged. The content of additional information messages depend on the utility used. **NOTE**: Verbose logging is in most cases not required and can lead to a decrease in performance.
`-nologappend`		If `-nologappend` is set, a new log file is created each time the utility is used with the same specification of `-logfile` and `-logpath`. The log file name is extended with a timestamp specifying the current UTC time. If `-nologappend` is not set, logging information is appended to the already existing log file each time the utility is used. **NOTE**: To restrict the file size, you can set the `-logclear` option to delete old log messages.
`-logclear` <number of days>	Infinite	This option can only be used if `-nologappend` is not set. If `-nologappend` is set, the `-logclear` setting is ignored. During logging, the log file is scanned for log messages with a timestamp older than the number of days specified with `-logclear` and these messages are deleted. **NOTE**: The scanning process can lead to drawbacks in performance.
`-heartbeat`	-1	Specification of the time between sending emails with the current content of the log file to the recipient specified with the option `-recipientmail`. If -1 is specified or a heartbeat is not defined, no email will be sent out and log information is exclusively available via the log file of the ADIF console application. **NOTE**: The email is meant as an alive message of the system informing the recipient about the availability of an active ADIF

Command Line Option	Mandatory/ Default	Explanation
		process. The content of the email is limited to the one-line content that is written to the log file at the moment the email is generated and is therefore not suitable for debugging. **NOTE**: The interval for sending the emails can also be specified with the **Debug Heart Beat** attribute of the ADIF scheme specified with the option-scheme or-schemefile. A specification in the command line overwrites a specification in the ADIF scheme. **NOTE**: The recipient's and sender's email address must also be specified either in the command line or in the ADIF scheme specified with the option-scheme or-schemefile to send information about the logging process via email.
-sendermail		Specification of the email address used as sender address in emails with log information sent out in the interval specified with-heartbeat to the recipient specified with-recipientmail. **NOTE**: The sender email address can also be specified with the **Sender Mail** attribute of the ADIF scheme specified with the option-scheme or-schemefile. A specification in the command line overwrites a specification in the ADIF scheme.
-recipientmail		Specification of the email address used as recipient address in emails with log information sent out in the interval specified with-heartbeat. **NOTE**: The recipient email address can also be specified with the **Recipient Mail** attribute of the ADIF scheme specified with the option-scheme or-schemefile. A specification in the command line overwrites a specification in the ADIF scheme.

Exporting Data to Microsoft® Word or PDF Format via the Alfabet Publication Framework

The Alfabet Publication Framework (APF) allows to publish current data from the Alfabet database to Microsoft® Word documents. Publication is done on basis of user defined Microsoft Word templates including bookmarks that are then mapped to relevant data of the Alfabet database via a publication configuration using the tool Alfabet Expand.

(i) For information about the required configuration of publications, see *Publishing Data In Microsoft® Word Format* in the reference manual *Configuring Alfabet with Alfabet Expand*.

Two different methods for triggering publications based on existing publication configurations are available. Both include functionality that must be triggered by a command line utility.

- Publications can be triggered by a user logged in to the Alfabet user interface. To trigger a publication, the user must open a customer defined report that either allows to publish data about an object the user selects in the report or to publish data about the object the user currently works with when opening the report via a button in the toolbar.

  The publications defined by users via the Alfabet user interface are stored in the database in the Alfabet Document Manager. The user can download the document either in the configured report or in a standard Alfabet functionality that lists all reports created during the last month. Documents that are older than one month are not displayed any longer on the Alfabet user interface but are still available in the Document Manager. Software AG provides a batch utility to delete old publications from the document explorer. It is recommended that you run this tool in regular intervals to clean the database from old publications.

  For more information about configuring the publication via the user interface, see *Publishing Data In Microsoft® Word Format* in the reference manual *Configuring Alfabet with Alfabet Expand*.

  For more information about the batch utility deleting the old documents from the database, see [Triggering Publication via a Batch Utility](#).

- Publications can be triggered by a command line utility. The resulting zip file containing the published Microsoft® Word documents is stored directly in the local file system. For information about publication via the command line utility, see [Deleting Expired Publications from the Database](#).

Triggering Publications via a Batch Utility

Software AG provides a Windows® command line tool PublicationConsole.exe to publish data on basis of a configured publication to the local file system.

Executable	`PublicationConsole.exe` located in the ***Programs*** subdirectory of the Alfabet installation directory
Remote or stand-alone access	Remote access with a remote alias connecting to a running Alfabet Server.
Preconditions	- An `AlfabetMS.xml` configuration file with a remote alias configuration specifying the connection to the running Alfabet Server is available. - A publication definition suitable for execution via a batch job is defined in Alfabet Expand and is in the **State** `Active`. For general information about APF and the required configuration of publications see *Publishing Data In Microsoft® Word Format* in the reference manual *Configuring Alfabet with Alfabet Expand*.
Logging	Log information is written to a log file and to the Alfabet database. Logging is configurable in the command line. For more information about logging, see [Standard Logging for Alfabet Batch Utilities](#).

| Command line help | Start executable with `-h` or `-help` |

Providing Relevant Data and Configuration

`PublicationConsole.exe` requires access to the following files and configurations:

A fully configured publication

A publication must be completely configured and set to the State `Active` before starting a publication job. The publication must be defined in the Alfabet database the Alfabet Server connects to and must be executable independent of selection of a base object. For information the required configuration of publications see *Publishing Data In Microsoft® Word Format* in the reference manual *Configuring Alfabet with Alfabet Expand*.

An AlfabetMS.xml configuration file

For access to the Alfabet database, an `AlfabetMS.xml` configuration file with an alias configuration must be available. The `AlfabetMS.xml` configuration file must contain a remote alias configuration to access the Alfabet Server that is connected to the Alfabet database. When starting the publication console application, the alias configuration name must be specified in the command line. If the `AlfabetMS.xml` configuration file containing the alias configuration is not located in the working directory of the publication console application, the path to the file must additionally be specified in the command line.

Starting the Publication Console Application

Start `PublicationConsole.exe` with the command line options listed in the table below.

You must start `PublicationConsole.exe` with at least

- a specification of the alias configuration defining access to an Alfabet Server or a Alfabet database,

- a specification of the user name for log in to the Alfabet Server,

- the name of the publication and,

- the target location for the published files in the local file system:

```
PublicationConsole.exe-msalias <alias name>-alfaLoginName <user
name>-alfaLoginPassword <user password>-publication <publication
name>-outputfile <file name>.zip
```

The table below displays the command line options:

Command Line Option	Mandatory/ Default	Explanation
`-msalias <alias name>`	Mandatory	Enter the remote alias name as specified in the `AlfabetMS.xml` configuration file. The corresponding Alfabet Server must be running.
`-msaliasesfile < Alfabet configura­tion file path>`	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName <user name>`	Mandatory	User name for access to the Alfabet database. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPassword <user password>`	Optional	Password for access to the Alfabet database.
`-publication <pub­lication name>`	Optional	Specify the name of the publication defined in Alfabet Expand.
`-outputfile <file name>`	Optional	Specify the name of and path to the ZIP file containing the publication output. **NOTE**: The extension of the file name must be.zip. The file with the defined name is created during the publication process and contains the published word document(s).
`-singledocument <true/false>`	Optional	This option is only relevant when data about multiple objects is published. When `-singledocument` is set to `true`, data about all selected objects is published in one document. Otherwise, one document is published per selected base object.
`-idletime`	Optional	The batch utility waits for alive messages from the Alfabet Server and if none are received within 60 seconds, the job is cancelled. The allowed wait time for alive messages can be changed by defining a new time interval in seconds with`-idletime`.
`-<parameter name> <parameter value>`	Optional	When the queries in the publication definition are configured with Alfabet query language parameters, the parameter can be specified in the command line. For each variable, a separate command line option must be specified as:

Command Line Option	Manda-tory/ Default	Explanation

		`-<parameter name> <value>`
		
		for example:
		
		`-fiscalyear '2009'`
		
		For more information about the use of variables, see Defining Publication Output in the Command Line of the Batch Utility.

Defining Publication Output in the Command Line of the Batch Utility

When a publication is triggered by `PublicationConsole.exe`, the selection of the base objects of the publication must be defined via a query in the publication definition. This results in a publication returning results for the same set of base objects each time it is executed.

If you want to select different base objects for each publication with PublicationConsole.exe, you can use Alfabet query language parameters in the queries selecting the base object as well as in the queries defining the data to be published about the objects. The value of the parameter is then defined in the command line of the publication console application when triggering a publication.

> (i) For more information about how to define parameters in Alfabet queries, see *Referring to the Current Alfabet Context in a WHERE Condition* in the chapter *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*.
>
> For more information about how to define parameters in native SQL queries, see *Using Alfabet Parameters* in the section *Defining Native SQL Queries* of the chapter *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*.

> The following configuration is required to use parameters in the command line:
>
> • When defining the publication, define queries that contain Alfabet query language parameters in `WHERE` conditions.
>
> • Optionally, define a default value for each parameter in the **Debug Arguments** attribute of the publication. The definition must be defined in the following syntax:
>
>   • Each default must be defined as <parameter name>=<value>.
>
>   • When multiple default values are added, they must be defined comma-separated.
>
>   • The <parameter name> does not include the prefix @ or: of the Alfabet query language parameter.
>
>   • The <value> must be written in the attribute as if defined in a query. For example, if a string is written in inverted commas in the query, the inverted commas are part of the value specification.
>
> • When starting the command line application, the value substituting the Alfabet query language parameter must be defined in the command line as-

<value>. The parameter name is the Alfabet query language parameter without the prefix @ or: and the value must be defined as required by the query syntax. This means that a string written in inverted commas in the query must be defined with the inverted commas.

A publication is defined that triggers the publication of data about a number of applications found via their name. The query finding the application is defined in the publication entry as:

```
ALFABET_QUERY_500

FIND Application

WHERE Application.Name LIKE:AppName
```

The attribute **Debug Arguments** of the publication definition specifies a default value for the application name. When no parameter value is defined in the command line, the publication will be created about all applications with a name starting with CMS:

```
AppName='CMS%'
```

When starting `PublicationConsole.exe` to create the publication about an application named "Central Server", the command line must be specified as follows:

```
PublicationConsole.exe-msalias Production-alfaLoginName SystemAdmin-
alfaLoginPassword AdminPassword-publication ApplicationData-
outputfile CentralServer.zip-AppName 'Central Server'
```

Deleting Expired Publications from the Alfabet Database

Software AG provides a Windows® command line tool `BatchExecutor.exe` to delete expired publications from the **Internal Document Selector** of the Alfabet database. This action is required if publications are generated by the users in the Alfabet user interface. Publications published via the command line tool `PublicationConsole.exe` are not stored in the **Internal Document Selector**.

Executable	`AlfaBatchExecutor.exe` located in the ***Programs*** subdirectory of the Alfabet installation directory
Remote or stand-alone access	Stand-alone access to the database with a server alias. Remote access with a remote alias connecting to a running Alfabet Server is only possible for the batch processing of monitors.
Preconditions	An `AlfabetMS.xml` configuration file with a server alias configuration specifying the connection to the Alfabet database is available and the respective functionality is configured.
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see [Standard Logging for Alfabet Batch Utilities](#).
Command line help	Start executable with-h or-help

The executable must be started with the following parameters:

```
alfaBatchExecutor.exe-jobClass ExpiredReportDeletionJob-msalias <alias
name>-alfaLoginName <user name>-alfaLoginPassword <user password>
```

The table below displays the command line options:

Command Line Option	Manda-tory/ Optional	Explanation
-jobClass <job name>	Manda-tory	Enter `ExpiredReportDeletionJob`.
-msalias <alias name>	Manda-tory	Enter the server alias name as specified in the `AlfabetMS.xml` configuration file. All other Alfabet applications with access to the Alfabet database must be stopped.
-msaliasesfile <Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the speci-fication of the alias is not located in the same directory as the ex-ecutable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
-alfaLoginName	Manda-tory	User name for access to the Alfabet database. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
-alfaLoginPassword	Optional	Password for access to the Alfabet database.

Exporting Data to XML, HTML, or Microsoft® Excel® Format with the QueryExecutor.exe

Software AG provides the command line tool `QueryExecutor.exe` for the export of object data from the Alfabet database to an XML, HTML or Microsoft® Excel® file. The Alfabet query that defines which data will be exported can be defined either directly in the export definition file or in a report configuration in Alfabet Expand.

⚠ `QueryExecutor.exe` does not work with native SQL queries. Alfabet query language must be used to define which data will be exported.

The following is required to batch export data to an XML, HTML or Microsoft® Excel® file.

If the data export is based on an Alfabet query defined in the export definition file:

- Creating an Export Definition for Export to XML, HTML or Microsoft® Excel® Files

- [Starting the Batch Export of Data to an XML, HTML or Microsoft® Excel® File](#)

If the data export is based on an Alfabet query defined in a report configuration in Alfabet Expand:

- [Creating an Alfabet Query-Based Report in Alfabet Expand](#)

- [Creating an Export Definition for Export to XML, HTML or Microsoft® Excel® Files](#)

- [Starting the Batch Export of Data to an XML, HTML or Microsoft® Excel® File](#)

In a typical scenario, export definitions and report configurations meeting the company's requirements are defined once and batch jobs for data export based on the existing export definitions are executed in regular intervals or on a per-demand basis.

Format of the Data Output

The output of the data export is a tabular report displaying the data found by the Alfabet query. Each column in the table represents a show property defined in the Alfabet query. For each result data set, a row is added to the report.

FIGURE: Example for export to HTML format

FIGURE: Example for export to Microsoft® Excel® format

In the XML file, the table is defined in a root element `AlfaDataSet` containing elements for the specification of columns and rows:

```
<AlfaDataSet>
    <AlfaDataSetCol>
        <Name>Reference</Name>
```

```
                    <Type>String</Type>
            </AlfaDataSetCol>
            <AlfaDataSetCol>
                    <Name>Domain.Name</Name>
                    <Type>String</Type>
            </AlfaDataSetCol>
            <AlfaDataSetCol>
                    <Name>Domain.CREATION_DATE</Name>
                    <Type>Date</Type>
            </AlfaDataSetCol>
            <AlfaDataSetRow>
                    <Reference>359-44-0</Reference>
                    <Domain.Name>Account Management</Domain.Name>
                    <Domain.CREATION_DATE>1/3/2008</Domain.CREATION_DATE>
            </AlfaDataSetRow>
        </AlfaDataSet>
```

For each column in the report, an XML element `AlfaDataSetCol` is added to the XML file, specifying the caption of the column in a child XML element `Name` and the data type in a child XML element `Type`.

> ⓘ The title of the column is given as <object class>.<property>. If an alias is specified for the column in the Alfabet query, the alias will be ignored for the XML output. In the HTML and Microsoft® Excel® output, the alias is used for the column caption.

For each row of the table, an XML element `AlfaDataSetRow` is added. It contains a child element for each column in the table defining the content of the table cell. The name of the child element is identical to the content of the `Name` element in the corresponding `AlfaDataSetCol` element.

Creating an Alfabet Query-Based Report in Alfabet Expand

The Alfabet query for the data export can be defined in a custom report in Alfabet Expand.

> ⓘ When you create an Alfabet query-based report, the report will not only be executable for batch data export, but can also be displayed to users in the Alfabet interface.
>
> Reports are only available on the Alfabet interface when they are in the state Active. Nevertheless, reports can be executed by the batch utility independent of the value defined for the State attribute of the report. This makes it possible to specify reports for batch execution without providing them to users in the Alfabet interface.

To create a new Alfabet query-based report in Alfabet Expand:

1) In the **Reports** tab of Alfabet Expand, right-click the **Reports** folder or a sub-folder and select **Create New Report**. You will see the new custom report listed below the selected folder.

As an alternative to defining all attributes for a new custom report, you can copy and paste an existing custom report and make modifications, as needed. To do so, right-click the existing custom report that you want to copy and select **Copy**. Right-click the new custom report and select **Paste**. Modify the necessary attributes described below.

2) In the property window, select **Query** in the **Type** field.

3) In the property window, define the following attributes for the report:

- **Name**: Enter a name for the report. The name you define here will be used to identify the report in your export definition for the batch utility.

- **Comment**: Optionally add information about the comment. If the state of the report is `Active`, the comment is also visible to users with access permissions to the custom report in the Alfabet interface.

- **State**: Select `Plan` to edit the report. For batch execution, the state of the report is not relevant.

  Custom report definitions are also used for by other mechanisms in Alfabet requiring additional attribute settings. For the definition of reports used for data export only the above mentioned settings are applicable.

4) Click the **Browse** button in the **Alfabet Query** attribute. In the dialog box that opens, select the base class for the Alfabet query. The **Query Builder** opens.

5) In the **Query** tab of the Alfabet Query Builder, define the Alfabet query for the selection of objects for the data export.

  For information about how to work with the Alfabet Query Builder, see the section *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*.

  The following is important for the generation of an Alfabet query for a report triggering data export:

- The use of parameters is restricted to the parameter:`TODAY`.

- You must not define filters for a report that is used for batch export of data.

- The specification of an alias in the **Show Properties** is ignored for export to XML.

6) In the **Show Properties** tab, select the attributes that are displayed in the columns of the report. In the **Sort Properties** tab, you can optionally select attributes for sorting the results alphabetically. Results are sorted in the order of specification of the sort properties.

7) Click **OK** to save the Alfabet query.

8) In the explorer, right-click the report and select **Test Report**. You can view the report in a preview window. If the results are not as expected, you can edit the Alfabet query until the output of the report meets your expectations.

  After upgrading to a new release of Alfabet, some of the Alfabet queries defined in reports might be affected by changes made to the meta-model or changes to the Alfabet query language. You can use the **Check and Update Reports** option in the context menu of either a single report, a report folder or the root node for reports in the explorer to test whether the queries in the relevant reports are syntactically correct. A report in Microsoft® Word® format will be created that provide information about which Alfabet

queries must be changed to work with the current version of Alfabet and to correspond to the current version of the Alfabet query language.

(i) The amount of data that can be displayed in a configured report is restricted in order to avoid exceeding memory size usage. If a configured report results in an error message informing you that the data set is too large, refine the query to return only a subset of the results. If you do not want to exclude objects from the configured report but rather would like to limit the set of data that is displayed, you can define filters that allow the user to restrict the data display as needed (for example, by displaying only objects with a name starting with a specific character or objects assigned to a specific ascendant object).

Creating an Export Definition for Export to XML, HTML or Microsoft® Excel® Files

The batch utility `QueryExecutor.exe` requires an export definition specifying which data will be exported and which separators will be used for export. To provide the export definition, create an XML file with a text editor that contains the definitions in an XML element `QueryExecutorDef:`.

(👁) The following example displays a configuration for the export of data based on a report defined in Alfabet Expand:

```
<QueryExecutorDef>

    <Query

        Name="ApplicationData"

        OutputType="HTML"

        ReportName="Application Data Export To HTML"

    />

</QueryExecutorDef>
```

The following example displays a configuration for the export of data based on an Alfabet query defined directly in the export definition.

```
<QueryExecutorDef>

    <Query

        Name="ApplicationData"

        OutputType="HTML"

        Query="ALFABET_QUERY_500 FIND Application

            LeftJoin BusinessFunction ON BusinessFunction.Domain =
            Domain.REFSTR

            SHOW Domain.Name BusinessFunction.Name

            SORT Domain.Name BusinessFunction.Name"

    />

</QueryExecutorDef>
```

The table below displays the attributes that can be edited for the XML element `QueryExecutorDef`.

AlfaExportDefinition Attribute	Mandatory/ Optional	Explanation
Name	Optional	The name of the output file. You should only use characters that are allowed in names for the output file format. If this parameter is not set, the output files are stored as Report<sequential number>.<file extension>. The location of the output file is specified in the command line options for the batch utility. For more information, see [Starting the Batch Export of Data to an XML, HTML or Microsoft® Excel® File](#).
OutputType	Mandatory	The format of the output type. Allowed values are: • XML (for XML output) • HTML (for HTML output) • XLS (for output to Microsoft® Excel®)
Query	Optional	Definition of data to be exported. The query must be written in Alfabet query language 500. The syntax of the Alfabet query language is described in the section *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*. **NOTE**: Either this attribute or either the ReportName or ReportID attribute must be specified to execute the export.
ReportName	Optional	Name of the custom report definition in Alfabet Expand that defines which data will be exported. **NOTE**: It is mandatory to specify either a report (via the ReportName attribute or the ReportID attribute) or an Alfabet query via the XML attribute Query to execute the export. **NOTE**: The report can be identified by either the ReportName attribute or the ReportID attribute.
ReportID	Optional	ID of the custom report definition in Alfabet Expand that defines which data will be exported. **NOTE**: It is mandatory to specify either a report (via the ReportName attribute or the ReportID attribute) or an Alfabet query via the XML attribute Query to execute the export. **NOTE**: The report can be identified by either the ReportName attribute or the ReportID attribute.

Starting the Batch Export of Data to an XML, HTML or Microsoft® Excel® File

The export of data will be triggered by a batch utility:

Executable	`QueryExecutor.exe` located in the **Programs** subdirectory of the Alfabet installation directory
Remote or stand-alone access	Stand-alone access to the database with a server alias or remote access with a remote alias connecting to a running Alfabet Server.
Preconditions	An AlfabetMS.xml configuration file with a server alias configuration specifying the connection to the Alfabet database is available and An export definition file was created. If the export definition file contains a reference to a custom report, the report must be defined in the tool Alfabet Expand.
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see Standard Logging for Alfabet Batch Utilities.
Command line help	Start executable with `-h` or `-help`

The following command line starts the batch job for data export with `QueryExecutor.exe`:

```
QueryExecutor.exe-msalias "<server or remote alias from
AlfabetMS.xml>"-alfaLoginName <user name>-alfaLoginPassword <user
login password>-xmlfile <name of the export definition file>-
outputdir <name of the target directory>
```

The table below displays the command line options:

Command Line Option	Mandatory/ Optional	Explanation
`-msalias` <alias name>	Mandatory	Enter the alias name as specified in the `AlfabetMS.xml` configuration file. If a remote alias is specified, the corresponding Alfabet Server must be running. If a server alias is specified, all other Alfabet applications with access to the Alfabet database must be stopped.
`-msaliasesfile` < Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.

Command Line Op- tion	Manda- tory/ Optional	Explanation
`-alfaLoginName` <user name>	Manda- tory	User name for access to the Alfabet database. ⚠️ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (`=True`) for the user.
`-alfaLoginPass- word` <user password>	Optional	Password for access to the Alfabet database.
`-xmlfile` <name of the export definition file>	Manda- tory	Name of the export definition file for data export, The name specifi- cation can contain the path to the file either absolute or relative to the working directory of the batch utility.
`-outputdir` <name of the target directory>	Manda- tory	Path to the target XML file, specified either absolute or relative to the working directory of the batch utility.

Exporting Data to XML Format with AlfaConsoleImportExport.exe

Software AG provides the command line tool `AlfaConsoleImportExport.exe` for the export of object data from the Alfabet database to an XML file. The tool is searching the database for relevant data via an export definition defined in the tool Alfabet Expand.

⚠️ `AlfaConsoleImportExport.exe` also includes a mechanism to import data from XML files to Alfabet. However, this mechanism only allows for the initial import of reference data and cannot be used to import object data for semantic classes to the Alfabet database. The data import functionality is only required for special processes that are not included in the system adminis- trator tasks for Alfabet. Therefore, this information is not included in this reference manual.

Software AG provides the Alfabet Data Integration Framework (referred to as ADIF) for batch im- port of data from XML files, CSV files, Microsoft Excel files, or external databases. For more infor- mation, see the reference manual *Alfabet Data Integration Framework*.

The following is required to batch export data to an XML file:

- Defining an Export Definition

- Starting the Batch Export of Data to an XML File

In a typical scenario, export definitions meeting the company's requirements are defined once and batch jobs for data export based on the existing export definitions are executed in regular intervals or on demand.

Format of the XML File Resulting from Export

The XML output of the export with `AlfaConsoleImportExport.exe` consists of the root XML element `AlfaXMLExport` with two child XML elements:

- `AlfaXMLExportHeader`: Contains basic information about the export (for example, the date and time that the export was performed).

- `AlfaInstances`: Contains the information about the objects chosen for publication.

The XML element `AlfaInstances` has the following child XML elements:

- `AlfaInstance`: Represents an object in the Alfabet database. One element is added for each object in the Alfabet database included in the report and can have the following child elements:

  - `AlfaPropertyValue`: Represents a property of the object. Except for the `AlfaPropertyValue` of the type `Text`, XML elements `AlfaPropertyValue` are empty and the properties of the objects are defined in the XML element's attributes.

- `AlfaRelations`: A relation table that specifies the relations between objects that are defined by properties of the type `Reference` or `ReferenceArray` in the publication. `AlfaRelations` contains the following child XML elements:

  - `AlfaRelation`: Represents a relation between objects. `AlfaRelation` elements give additional information about relations specified in object properties of the type `Reference` or `ReferenceArray`.

The following example displays the structure of XML elements in the publication output. XML elements are not repeated in the example below although this is allowed for most XML elements.

```xml
<?xml version="1.0" encoding="utf-8"?><AlfaXMLExport>

    <AlfaXmlExportHeader FullDatabase="false" Comment=""
    Exported="2009-03-03T11:13:45.8030211" Classes="0" Views="0"
    Enums="0" Instances="0" />

    <CssFile PATH="DefaultCssCSS" />

    <AlfaInstances>

        <AlfaInstance ClassName="Application" Reference="76-2789-0">

            <AlfaPropertyValue Name="ID" Type="String" Datum="APP-
            2789" />

        </AlfaInstance>

        <AlfaRelations>

            <AlfaRelation FromRef="76-2789-0" Property="Documents"
            ToRef="34-280-0" />

        </AlfaRelations>

    </AlfaInstances>

</AlfaXMLExport>
```

The following XML elements and their XML attributes define the exported data from the Alfabet database:

XML Element (bold)/XML Attribute	Meaning:
AlfaInstance	**Represents an Object in the Alfabet database**
ClassName	The Alfabet object class the object belongs to.
Reference	The reference string that is used to identify this object in the export XML file. The reference string is used in properties of the type `Reference` or `ReferenceArray` or in the `AlfaRelation` child XML elements to specify this object.
AlfaPropertyValue	**Represents a property of an object.**
Name	The name of the property. For example: `Name`, `ID`, or `ResponsibleUser`
Type	The data type of the property in the database. For example: `String`, `Text`, `Reference`, or `ReferenceArray`
Datum	The value of the property. Properties of the type `Reference` or `ReferenceArray` represent references to other objects. Their attribute value displays the value of the `Reference` attribute of the referenced object(s). Properties of the type `Text` does not contain an attribute `Datum`. Instead, these `AlfaPropertyValue` elements contain content of the type `CDATA`.
AlfaRelation	**Represents a reference between two objects**
FromRef	Reference string of an object specified in an XML element `AlfaInstance` of this publication XML containing a reference to another object in a property of the type `Reference` or `ReferenceArray`.
Property	Name of the object class that is referenced.
ToRef	Reference string of the object that is referenced.

Only properties that are set for an XML attribute are specified in the output XML file. For example, if the property **ShortName** is exported for applications, the XML element `AlfaPropertyValue` with the XML attribute `Name="ShortName"` is only available in the specification of an XML element `AlfaInstance` of the class **Application** when a short name is defined for that application.

Properties of the Type String, Boolean, Date

Output in the XML publication:

```
<AlfaInstance ClassName="Application" Reference="76-2789-0">

    <AlfaPropertyValue Name="ID" Type="String" Datum="APP-2789" />

</AlfaInstance>
```

The value of the property is directly specified in the XML element `AlfaPropertyValue` with the XML attribute `Datum`.

Dates are written in ISO 8601 format as "year-month-dayThour:minutes:seconds.fraction". For example: 2005-04-15T12:46:47.577.

Properties of the Type Text

Output in the XML publication:

```
<AlfaInstance ClassName="Application" Reference="76-2789-0">

    <AlfaPropertyValue Name="Description" Type="Text">

        <![CDATA(Implementation of further functionalities from
        project 287)]>

    </AlfaPropertyValue>

</AlfaInstance>
```

The XML attribute `AlfaPropertyValue` with `Type="Text"` has no XML attribute `Datum`.

The value of the property is specified as content of the XML attribute `AlfaPropertyValue` in a child XML element in CDATA format.

Properties of the Type Reference or ReferenceArray

Output in the XML publication:

```
<AlfaInstance ClassName="Application" Reference="76-2789-0">

    <AlfaPropertyValue Name="LocalComponents" Type="ReferenceArray"
    Datum="175-5519-0,175-5520-0,175-5521-0" />

</AlfaInstance>

<AlfaRelations>

    <AlfaRelation FromRef="76-3411-0" Property="LocalComponents"
    ToRef="175-5519-0" />

    <AlfaRelation FromRef="76-3411-0" Property="LocalComponents"
    ToRef="175-5520-0" />

    <AlfaRelation FromRef="76-3411-0" Property="LocalComponents"
    ToRef="175-5521-0" />
```

```
</AlfaRelations>
```

The value of the property is directly specified in the XML element `AlfaPropertyValue` with the XML attribute `Datum`. The XML attribute `Reference` of the XML element `AlfaInstance` of the referenced object is defined as reference string. For reference arrays, multiple reference strings can be defined in comma-separated format.

For each reference string in the XML attribute `Datum`, an XML `AlfaRelation` is added to the output that defines the object class that is referenced, but no additional information about the referenced object.

In the output file, the XML attribute `Reference` used in the XML element `AlfaInstance` is used for references instead of the `REFSTR` property of the referenced object. Therefore, it is not possible to define which object is referenced unless the object itself is also included in the output with an `AlfaInstance` definition. For information about how to include related objects in the published output, see Defining Classes Referenced by Properties of the Base Class.

From the example below, you can see that after adding the referenced objects to the export definition, the publication XML output includes all information required to generate an XSL that substitutes the reference strings in the XML attribute `Datum` of the XML element `AlfaPropertyValue` with the name of the referenced object.

Output in the XML publication:

```
<AlfaInstance ClassName="Application" Reference="76-2789-0">

    <AlfaPropertyValue Name="LocalComponents" Type="ReferenceArray"
    Datum="175-5519-0,175-5520-0,175-5521-0" />

</AlfaInstance>

<AlfaInstance ClassName="Application" Reference="76-2789-0">

    <AlfaPropertyValue Name="LocalComponents" Type="ReferenceArray"
    Datum="175-5520-0,175-5521-0" />

</AlfaInstance>

<AlfaInstance ClassName="LocalComponent" Reference="175-5521-0">

    <AlfaPropertyValue Name="Name" Type="String" Datum="CRM" />

</AlfaInstance>

<AlfaInstance ClassName="LocalComponent" Reference="175-5520-0">

    <AlfaPropertyValue Name="Name" Type="String" Datum="Access" />

</AlfaInstance>

<AlfaRelations>

    <AlfaRelation FromRef="76-3411-0" Property="LocalComponents"
    ToRef="175-5520-0" />

    <AlfaRelation FromRef="76-3411-0" Property="LocalComponents"
    ToRef="175-5521-0" />

</AlfaRelations>
```

Properties with Links to Attachments

Output in the XML publication:

```
<AlfaInstance ClassName="Application" Reference="76-2789-0">

    <AlfaPropertyValue Name="Documents" Type="ReferenceArray"
    Datum="34-280-0" />

</AlfaInstance>

<AlfaInstance ClassName="ALFA_IDOCUMENT" Reference="34-280-0"
NAME="Application.zip" PATH="TestFolder\Application.zip" />

<AlfaRelations>

    <AlfaRelation FromRef="76-2789-0" Property="Documents" ToRef="34-
    280-0" />

</AlfaRelations>
```

The value of the property is directly specified in the XML element `AlfaPropertyValue` with the XML attribute `Datum`. The XML attribute `Reference` of the XML element `AlfaInstance` of the referenced object is defined as reference string. Multiple reference strings can be defined in comma-separated format.

For each reference string in the XML attribute `Datum`, an XML `AlfaRelation` is added to the output that defines the object class that is referenced, but no additional information about the referenced object.

For a meaningful publication of documents and attached Web links, it is required that the property **Is Publication** of the export definition is set to `True` and the property **Integrity Reference Properties** of the class definition in the export definition includes Documents and WebLinks. Only when this setting is done, an XML element `AlfaInstance` is generated in the publication XML file for each attached document and Web link. These XML elements `AlfaInstance` have an additional attribute PATH, which specifies the following:

- For WebLinks: the target of the link.

- For Documents: The location of the documents. Attachments are exported together with the publication and are located in a sub-folder of the location where the publication is stored.

Defining an Export Definition

Export definitions provide information about which data will be exported to an XML file. The specification of objects and attributes relevant for export is done using the Alfabet query language.

Export definitions are defined with the tool Alfabet Expand.

To create an export definition:

1) In the **Exp/Imp** tab of Alfabet Expand, right-click **Export** and select **New Export Definition**. You will see an object labeled **New Definition** listed as the last entry in the **Export** tree.

2) In the property window, define the following properties for the export definition:

- **Name**: Define a unique name for the export definition.

- **Comment**: Optionally enter comments providing information about the export definition.

- **Is Publication**: Select `True` if documents attached to an objects should be exported to a directory Publications in the output directory for the export results and Web links attached to objects will be included in the results. If Web links and documents will not be included in the export, the property **Is Publication** is not relevant.

> (i) Export definitions are also used by other mechanisms in Alfabet requiring additional attribute settings. For the definition of export definitions for data export to XML files only the above mentioned settings are applicable.

3) In the toolbar, click the **Save** button to save your changes.

Adding a Class to the Export Definition

Classes are defined for an export definition for the following purposes:

- Every export definition requires a base class, which is the artifact that determines the content of the export. The base class is the first class that you must define and should be at the top of the list of classes for your export definition.

- When a property of the base class of the type Reference or Reference Array is included in the publication, the XML output of the publication will only allow to identify the referenced objects when the property is specified as an ***Integrity Reference Property*** and the class is added to the publication in the way described in the section Defining Classes Referenced by Properties of the Base Class.

- If a class should be added to the report that is not referenced by a property of the base class, an Alfabet query can be defined as child element of the base class definition to specify the relation between the classes (for example, if local components are the base class and the application that the local component is defined for will be added to the report). The relation between applications and local components is defined in a property of the class Application. Therefore, an Alfabet query must be defined that searches for applications with a reference to any of the local components in the report. The class must be added to the export definition as described in section Defining Classes Referenced by Subordinate Alfabet Queries of the Base Class.

The classes that are defined with a relation to the base class can also reference new classes, either by definition of a subordinate Alfabet query or because of a published property defining a reference. The classes related to these classes must also be specified in the export definition.
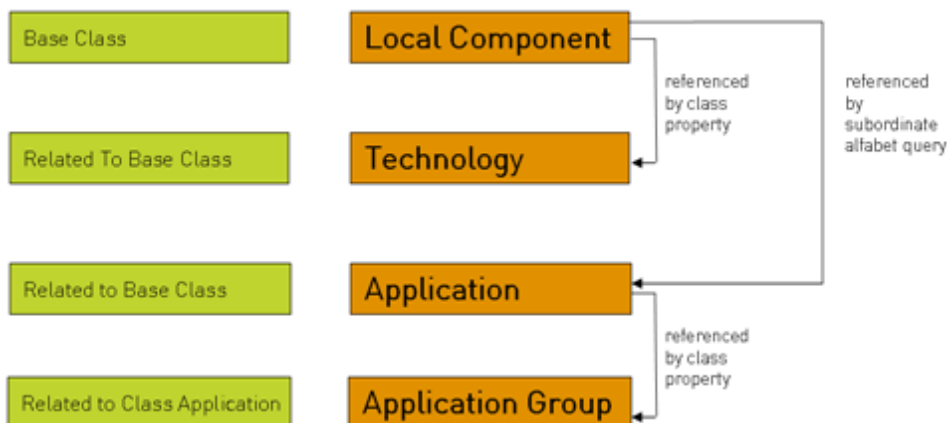
FIGURE: Example for relations between classes defined in an export definition

In the following, the steps to include a new class in the publication are described for the base class only, but you can follow the described procedures also to add classes related to any other class defined in the export definition. In this case, substitute base class in the description with the class that you want to define the related class for.

Adding a Class as a Base Class of the Export

The definition of classes for the export definition must start with the base class (which is the main object class for the data that will be published). A publication can have only one base class.

To add a base class to the export definition:

1) Right-click the export definition that you want to add a class to and select **Add Class**. The **Select Class** dialog box opens.

2) In the dialog box, click the class that you want to add to the export definition and click **OK**. You will see the new class under its export definition. In the property window, define the following properties for the class:

   • **Query**: Define an Alfabet query to gather the content from the Alfabet inventory for the export. No show and sort properties must be specified for the Alfabet query. The properties that are included in the publication are selected in the property window of the class definition instead.

     ⓘ For information about how to define Alfabet queries, see *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*.

   • **Properties to Export**: Click the **Browse** ▦ button, select the properties for the class to be included in the export and click **OK**.

   • **Integrity Reference Properties**: You must specify all properties of the base class selected in the **Properties to Export** field for which one of the following conditions apply.

     • The properties Documents or WebLinks of any object class. The documents are then exported with the documentation and information about the location of the exported documents and the target of Web links is added to the export. For information about the format of the information in the resulting publication output, see the section Format of the XML File Resulting from Export. If you do not specify **Integrity Reference Property**, you can only see from the export whether documents or Web links are specified for the objects without information about their content.

     • The property specifies a reference to another object class. This applies to all properties of the data type `Reference` and `ReferenceArray`.

       ⓘ If your selection of properties include properties that are references to other classes, you must also define the referenced class as described in the section Defining Classes Referenced by Properties of the Base Class.

       ⓘ If your selection of properties includes WebLinks or documents, the property **Is Publication** of the export definition must be set to `True`.

3) In the toolbar, click the **Save** button to save your changes.

Defining Classes Referenced by Properties of the Base Class

If you want to include a property of the base class in your publication that specifies a reference to other objects classes, you must define the following:

1) In the explorer, click the base class in your export definition to open the property window.

2) Select the property that defines the relation to the other object class in both the **Properties to Export** and **Integrity Reference Properties** property.

3) In the explorer, right-click your export definition and select **Add Class**. The **Select Class** dialog box opens.

4) Select the class that you want to add to the export definition and click **OK**. You will see the new class under its export definition.

5) In the property window, delete the Alfabet query from the **Query** field. (The Alfabet query was automatically added to the class definition).

> ⚠ If you do not delete the Alfabet query, your publication will include all available objects of the defined class and not only the objects related to your base class.

6) Define the other properties of the new class. For a description of available properties, see the section Adding a Class as a Base Class of the Export.

> ⓘ A property can reference objects of the same object class than the base class to define a hierarchical relationship. For example, the class `ApplicationGroup` has an attribute `BelongsTo` that references subordinate application groups. If you want to define the hierarchy within an object class, you must define the relevant property of the class in the **Properties to Export** and **Integrity Reference Properties** properties of the base class only. You cannot add a second definition of the class to the export definition. As a result, all hierarchical levels of the object class starting from the objects defined by the base Alfabet query are included in the publication.

Defining Classes Referenced by Subordinate Alfabet Queries of the Base Class

In some cases, you may want to include information that is relevant to the base class even though the base class has no direct link to the information. You can define an Alfabet query to traverse from the export definition's base class to another class in the meta-model that has the content you want to include in your export. For example, you could specify an Alfabet query from the class Component Group to the class Aspect Indicator.

If you want to include a property of the base class in your publication that specifies a reference to other objects classes, you must define the following:

1) In the explorer, right-click the base class in your export definition and select **New Query**. The **Select Class** dialog box opens.

2) In the dialog box, click the class that you want to add to your publication and click **OK**. The Alfabet Query Builder opens and shows the selected class as base class.

3) Define the Alfabet query to define which objects of the selected class should be published and click **OK** to close the Alfabet Query Builder. To define the relation to the base class, you must define a `WHERE` clause that specifies the relation with the base class objects. Use the parameter:`BASE` in the `WHERE` clause to refer to the `REFSTR` attribute of any object of the base class included in the publication.

4) In the explorer, right-click your export definition and select **Add Class**. The **Select Class** dialog box opens.

5) In the dialog box, click the class that you want to add to the export definition and click **OK**. You will see the new class under its export definition.

6) In the property window, delete the Alfabet query from the **Query** field. (The Alfabet query was automatically added to the class definition).

> ⚠️ If you do not delete the Alfabet query, your publication will include all available objects of the defined class and not only the objects related to your base class.

7) Define the other properties of the new class. For a description of available properties, see the section [Adding a Class as a Base Class of the Export](#).

Starting the Batch Export of Data to an XML File

The export of data will be triggered by a batch utility:

Executable	`AlfaConsoleImportExport.exe` located in the **Programs** subdirectory of the Alfabet installation directory
Remote or stand-alone access	Stand-alone access to the database with a server alias or remote access with a remote alias, connecting to a running Alfabet Server.
Precondi-tions	An `AlfabetMS.xml` configuration file with a server alias configuration specifying the connection to the Alfabet database is available and an export definition was defined in the tool Alfabet Expand. For information about creating export definitions, see the section [Defining an Export Definition](#).
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see [Standard Logging for Alfabet Batch Utilities](#).
Command line help	Start executable with `-h` or `-help`

> 👁 The following command line starts the batch job for data export with `AlfaConsoleImportExport.exe`:
>
>   AlfaConsoleImportExport.exe-msalias "<server or remote alias from
>   AlfabetMS.xml>"-alfaLoginName <user name>-alfaLoginPassword

```
login password>-exportdef <name of the export definition>-e <name of
the target XML file>
```

The table below displays the command line options:

Command Line Option	Manda-tory/ Optional	Explanation
`-msalias` <alias name>	Manda-tory	Enter the alias name as specified in the `AlfabetMS.xml` configuration file. If a remote alias is specified, the corresponding Alfabet Server must be running.
`-msaliasesfile` <Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName`	Manda-tory	User name for access to the Alfabet database. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPassword`	Optional	Password for access to the Alfabet database.
`-exportdef` <name of the export definition>	Manda-tory	Name of the export definition for data export that is the value of the **Name** attribute of the export definition.
`-e` <name of the target XML file>	Manda-tory	Name and path to the target XML file.
`-disposeafterquery` <true/false>	Optional	The default for this parameter is either a preconfigured value or "`false`".

Exporting Data in Value-Separated Format (CSV)

Software AG provides the tool `AlfaExportUtil.exe` to batch export data from the Alfabet database to an export file in which exported data is separated by specified separators. Separators can be defined for the separation of object records and for the separation of property values exported per object.

If a separator is used within an exported object or attribute (for example, as part of the name), `AlfaExportUtil.exe` will either ignore that character or replace it with a preconfigured character.

In the example, data about applications is exported. For each application, the ID, name, version and the start and end date are written to the file.

Commas are used to separate property values and a line break is used to separate object records:

```
APP-57,CMS-System,1.2,1/1/2008,1/1/2010

APP-66,PublicationSystem,2.4,24/6/2007,1/6/2009

....
```

The following is required to batch export data in comma-separated format:

- [Creating an Export Definition for Export in Comma-Separated Format](#)

- [Starting the Batch Export of Data to a Comma-Separated File](#)

In a typical scenario, export definitions meeting the company's requirements are defined once and batch jobs for data export based on the existing export definitions are executed in regular intervals or on demand.

Creating an Export Definition for Export in Comma-Separated Format

The batch utility for data export to a comma-separated format file requires an export definition specifying which data should be exported and which separators should be used for export. The batch utility reads the export definition from an XML file that must contain the definitions in an XML element `AlfaExportDefinition`.

The following example displays a configuration for the export of data for the class ICT Objects:

```
<AlfaExportDefinition

    MSAliasFile="AlfabetMs.xml"

    MSAlias="Alfabet30"

    ClassName="Application"

    ExportAttributes="ID,Name,Version,StartDate,EndDate"

    SortAttributes="ID,Name"

    AQLWhere="(AND StartDate &gt; '01.01.2007' EndDate &lt;
    '01.01.2012')"

    ExportFile="TestExport.csv"

    ValueSeparator="," RecordSeparator="\r\n"

    ReplaceValueSeparatorWith=";"

    ReplaceRecordSeparatorWith=" "

    SavePreviousExportFile="true"

    Encoding="unicode"

/>
```

The table below displays the XML attributes that can be edited for the XML element `AlfaExportDefinition`.

XML Attributes for XML Element AlfaExportDefinition	Mandatory/ Optional	Explanation
MSAliasFile	Optional	Name of the Alfabet configuration file of the Alfabet Server that has access to the database. Typically this is the `AlfabetMS.xml` file. This parameter can also be specified in the command line for starting the batch utility. If it is defined in both the export definition and the command line, the definition in the command line is ignored.
MSAlias	Optional	Alias name of the Alfabet Server that has access to the database. This parameter can also be specified in the command line for starting the batch utility. If it is defined in both the export definition and the command line, the definition in the command line is ignored. If it is not defined in the command line, the definition in the export definition is mandatory.
ClassName	Mandatory	Name of the Alfabet object class for which the data should be exported.
ExportAttributes	Mandatory	Comma-separated list of the names (**not the captions!**) of the object properties that should be exported. Property values are written to the records in the output file in the order specified here.
SortAttributes	Optional	Comma-separated list of the names (**not the captions!**) of class properties used to sort the object records in the output file. Sorting is done in the order of specification.
AQLWhere/ DirectSql	Mandatory	Data for export must be specified by defining either of the following: • A `WHERE` statement entered in Alfabet query language 500 can be defined in the XML attribute `AQLWhere`, finding a subset of the objects of the object class specified with the attribute `ClassName`. • A native `SQL` query with a `SELECT` statement finding objects of the object class defined with the XML attribute `Class Name` can be defined in the XML attribute `DirectSql` The syntax of the Alfabet query language is described in the section *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*.

XML Attributes for XML Element AlfaExportDefinition	Mandatory/ Optional	Explanation
ExportFile		Name of the export target file. The specification to the file location can be either an absolute or relative path to the working directory of the batch utility.
		If the file does not exist, it will be created during export. If it already exists, it will be overwritten, but old content can be written to an archive file prior to writing the new data to the file. Archiving is activated via the **SavePreviousExportFile** attribute which is explained below.
ValueSeparator		String to separate values of exported object properties in the export file. You can use the following system characters in the separator string: The separator string will be parsed for system characters and changed accordingly.
		\t = tabulator
		\n = new line
		\r = new paragraph
RecordSeparator		String to separate object records in the export file. You can use the following system characters in the separator string: The separator string will be parsed for system characters and changed accordingly.
		\t = tabulator
		\n = new line
		\r = new paragraph
ReplaceValueSeparatorWith		String to replace the value separator string in case that string is found in the exported data. If "ReplaceRecordSeparatorWith" is not set, the record separator string will be ignored.
ReplaceRecordSeparatorWith		String to replace the record separator string in case the string is found in the exported data. If "ReplaceRecordSeparatorWith" is not set, the record separator string will be ignored.
SavePreviousExportFile		Enter True if you want an existing previous export file with the same name to be saved as <export file name><current data and time><export file name extension> prior to export of the data, or False if you want to overwrite previous data. Default is True.
Encoding		Character encoding for the export file. Default is Unicode.

Starting the Batch Export of Data to a Comma-Separated File

The export of data will be triggered by a batch utility:

Executable	`AlfaExportUtil.exe` located in the **Programs** subdirectory of the Alfabet installation directory
Remote or stand-alone access	Stand-alone access to the database with a server alias or remote access with a remote alias, connecting to a running Alfabet Server.
Preconditions	An export definition file is available. The export definition file must be defined as described below.
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see Standard Logging for Alfabet Batch Utilities.
Command line help	Start executable with `-h` or `-help`

👁 The following command line starts the batch job for data export with AlfaExportUtil.exe:

```
AlfaExportUtil.exe- f <export definition file>-alfaLoginName <user
name>-alfaLoginPassword <user login password>
```

The table below displays the command line options:

Command Line Option	Mandatory/Optional	Explanation
`-msalias <alias name>`	Mandatory if not specified in the export definition file	Enter the alias name as specified in the `AlfabetMS.xml` configuration file. If a remote alias is specified, the corresponding Alfabet Server must be running. This parameter can also be specified in the export definition file. If it is defined there, a definition in the command line is ignored. If it is not defined in the export definition file, the definition in the command line is mandatory.
`-msaliasesfile <Alfabet configuration file path>`	Optional	If the `AlfabetMS.xml` configuration file that contains the specification of the alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter. This parameter can also be specified in the export definition file. If it is defined there, a definition in the command line is ignored. If it is not defined in the export definition file, the definition in the command line is used.

Command Line Option	Mandatory/Optional	Explanation
`-alfaLoginName` <user name>	Mandatory	User name for access to the Alfabet database. ⚠️ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPass-word` <user password>	Optional	Password for access to the Alfabet database.
`-f` <name of the export definition file>	Mandatory	Name (including path) of the export definition XML file

Accessing the Alfabet Database with External Applications

The Alfabet database allows for an external SQL-based search to be executed. No Alfabet components are required for external database access. As a consequence of this, Alfabet access permissions and mandate management are not taken into consideration. Only the management of access permissions to the database server apply.

⚠️ For security reasons, the access to the Alfabet database for external reporting should not be an administrator access. ReadOnly rights should be granted in order to avoid database corruptions or data loss due to changes performed on the Alfabet database.

This method allows you to generate reports based on up-to-date data in your Alfabet inventory and make the reports available in the reporting environment (for example, in a reporting tool or in the intranet by means of a dynamic web page) independent of the design of the standard Alfabet reports. The design and query language used to build the report are specific to the external reporting tool and not described in this documentation.
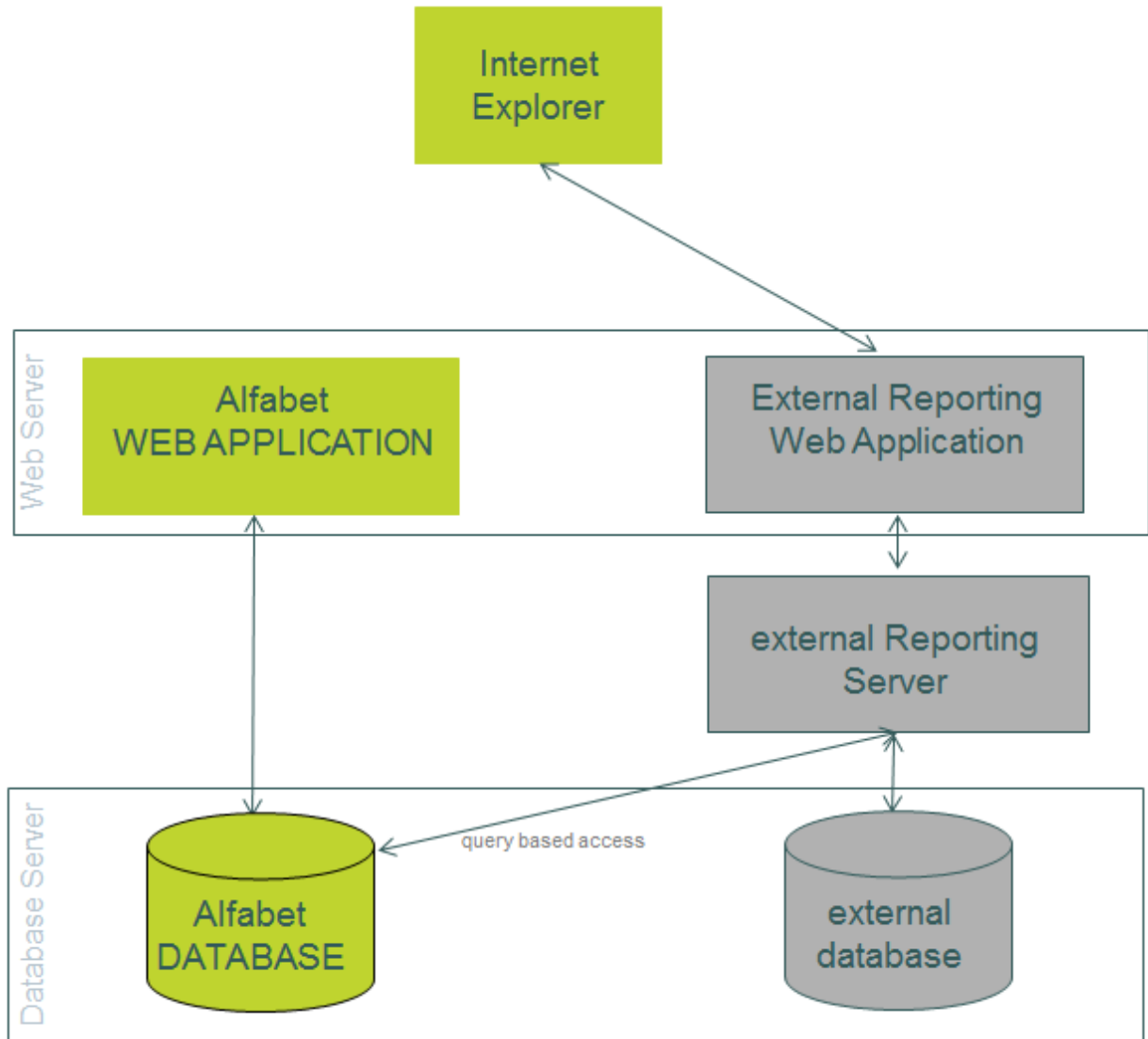
FIGURE: External reports based on third-party query-based access to the Alfabet database

In the figure above, the reporting tool provides the reports to end users via Web access. The external reporting tool retrieves data from the Alfabet database and can process data from another database as well. The external reporting tool functions independent of the Alfabet Web Application.

The user accesses the Web application of the external reporting tool with the Web browser. The external reporting tool generates the requested report from the data retrieved via the SQL-query based access to the Alfabet database.

The following figure shows an external report about the strategic alignment of the business support provided by the applications of different organizations within the company. The report lists all organizations in the order of their evaluation average. For each organization, the average evaluation for each evaluation type is displayed in the table and graphic information shows the average value as a horizontal bar and the range as a green vertical line.
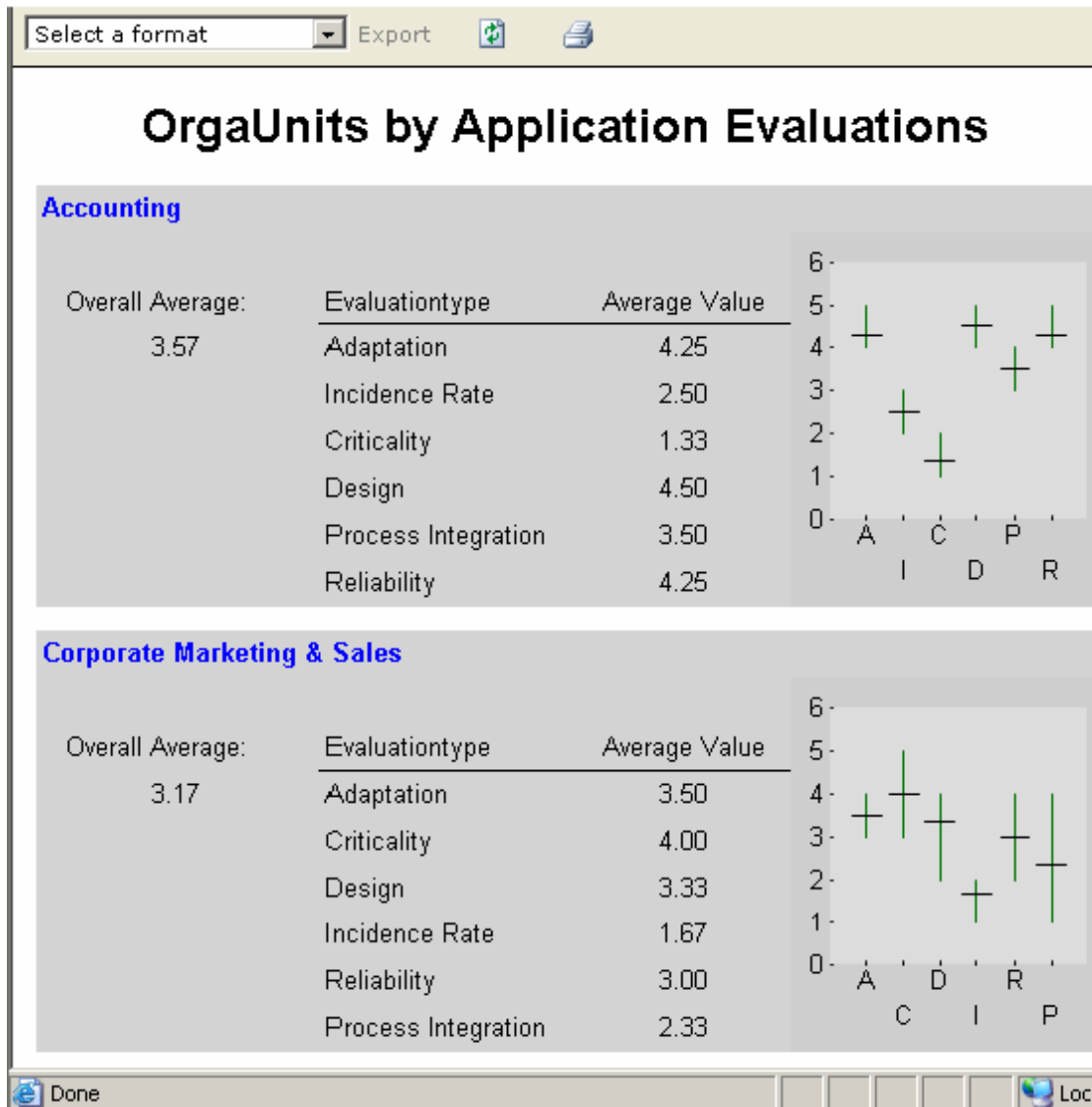
FIGURE: External report about business alignment of applications owned by different organizations

The report displays the data from the Alfabet database. The reporting tool reads the name of the organizations and which applications are owned by the organizations from the database table for organizations (`ORGAUNIT`). The reporting tool also reads the evaluations for the organization's applications from the `APPLICATIONS` database table and calculates the average values and ranges in the report from the evaluation values.

The reporting tool is responsible for data collection from the database, calculating and sorting values, and the graphic visualization of the data.

The mechanisms required for the reporting tools to access the Alfabet database will depend on the reporting tool and database server used. These mechanisms are therefore not documented in this manual but should be described in the documentation of the relevant external tool.

In addition to direct access to the Alfabet database, interfaces for information interchange between the Alfabet user interface and Web-based external applications are available and described in this chapter:

- [Links to Alfabet Views from External Applications](#)

  Software AG provides a link syntax that allows a specified view of the Alfabet interface to be opened with up-to-date data by clicking on the link in the external application. For example, the links can be integrated into reports generated by external reporting tools, the Intranet, a portal or Web site or an email. Access permissions can be configured to ensure security.

- [Integration of External Reports into Alfabet](#)

  Links to external URLs can be integrated into the Alfabet interface, allowing the Alfabet user to access information from other data sources without leaving the working environment. Not only does this mechanism allow for the integration of information about external data to Alfabet, but also the generation of customized reports about the data in the Alfabet database with an external reporting tool. This information can be provided in addition to the standard Alfabet report page views in the Alfabet user interface.

O/R Mapping Information Relevant for SQL-Based Access

The following information is limited to the description of the database fields required to access the Alfabet database with external tools.

The name of the database table is specified by the **Tech Name** attribute of the object class. The names of the columns in the table are specified by the **Tech Name** attributes of the object class properties of the respective object class. For most standard Alfabet object classes and properties, the **Tech Name** is identical to the **Name** attribute of the object class or property. All **Tech Name** attributes are stored in upper case letters in Alfabet (<TECHNAME>).

If the technical name derived from the object class name conflicts with keywords reserved for the relational database management system (RDBMS) of the Oracle or Microsoft SQL Server , the **Tech Name** will be written with a prefix "T_" for tables of object classes and a prefix "A_" for columns representing attributes.

> ⚠️ Reserved keywords are different for Oracle® and Microsoft® SQL® relational database management systems. Therefore, the migration of a Alfabet database between these environments may fail due to conflicts between the TECHNAME attribute and the reserved keywords of the new relational database management system. Before the migration process is executed, the TECHNAME specifications must be checked for keywords reserved for the new relational database management system and altered, if necessary.

For custom attributes defined by your company with the tool Alfabet Expand, the **Tech Name** attribute must be explicitly defined. Otherwise database migration procedures will not work. For more information about defining the **Tech Name** attribute, see the section *Configuring Custom Properties for Protected or Public Object Classes* in the reference manual *Configuring Alfabet with Alfabet Expand*.

When the **Support Data Translation** attribute is set to True for a language definition in Alfabet, the Name and Description properties of the objects in the database can be translated to the language culture. When a property is translatable, the translation is written to the database table column "<property tech name>_<language code>"

REFSTR	INSTGUID	ID	NAME	NAME_1031	DES
76-2518-0	117F79F6CC694...	APP-2518	Business EAI Pla...	NULL	NULL
76-2525-0	C88B1BFE80A24...	APP-2525	Groupware Servi...	NULL	NULL
76-2538-0	9447219E743F4...	APP-2538	Mafo-Portal	NULL	NULL

FIGURE: Section of a database table for applications with column headers specified with the TECHNAME property

The `REFSTR` property is a unique internal object ID that explicitly identifies each object in the database and is used for references between objects in the database.

> The `REFSTR` property shall not be used in queries to identify single objects. Instead, it is recommended that you use unique key attributes such as an object's name and version. If you archive a database and restore the archive in another database, the `REFSTR` property of single objects may change. This is not affecting references between objects. References are automatically adapted to the new `REFSTR` scheme.

Data for the Alfabet property types `Text` (for example, object descriptions, attributes of the type `StringArray`) and `Date` are stored differently in different database servers:

Database	Data type: Text	Data type: Date
Oracle 9/10/11	NCLOB	TIMESTAMP
Microsoft SQL 2005/2008	nvarchar(max)	datetime

These data types may require special handling or possibly may not be queried from a certain reporting tool.

Object properties of the type `ReferenceArray` are handled in two different ways depending on the specification of the **Reference Support** attribute of the property:

- If **Reference Support** is set to `True`, the relations specified by the property are stored in the `RELATIONS` table. No database column is available for the property in the table of the object itself. The `RELATIONS` table specifies the relation in the following database columns:

  - `FROMREF`: The `REFSTR` value of the object that is referencing another object.

  - `PROPERTY`: the property of the object specified with `FROMREF` that defines the relation between the objects.

  - `TOREF`: The `REFSTR` value of the referenced object

- If **Reference Support** is set to false, the relations are stored directly in the database table of the object as a string. The `REFSTR` of all objects referenced by the property are listed whitespace separated.

Links to Alfabet Views from External Applications

External applications can link to Alfabet page views, object profiles, configured reports, explorers and business functions. If a user clicks the link to Alfabet in the external report, the Alfabet interface will open in the Web browser and display the referenced view. The links to the Alfabet views require a specific syntax. For more information, see Link Syntax for Links to Alfabet Views from External Applications.

The following applies to Alfabet views opened from an external application:

- Only page views for which bookmarks can be set can be accessed from external applications.

- If a standard view has filters to specify its content, the filters will not be automatically set when opening the view from the external application. The user accessing the view must specify the filter settings.

- If a configured report requires specification of filter settings, the required settings can be included in the link and the configured report opens with the filter settings defined in the link. If the configured report is configured not to open immediately but only after the user clicked the Submit button, the filter settings will be preset, but the report is only displaying results after the user actively clicked the **Submit** button.

- If an explorer shall open, the path to the explorer node that shall be selected on opening the explorer can optionally be defined in the link.

- The user profile used to display the view is specified in the link. If no user profile is specified, the default user profile for anonymous users is used.

  The presentation of data may change depending on the user profile used for log on. For example, it is possible to hide page views that are available by default for an object.

- Access is determined by the parameters defined in the link as well as the activation of the external access capability in the AlfabetMS.xml configuration file. For more information, see Managing Access Permissions for Access from External Applications.

In the figure above, the user working with the external reporting tool will begin working in the environment of the reporting application. This is the Web application of the external reporting tool. When the user clicks a link to the Alfabet interface, the Web browser retrieves the view from the Alfabet Web Application. The configuration of the reporting application will determine whether the Alfabet Start screen opens in a new window, in the current window (which would lead away from the external reporting tool), or in a frame of the reporting application. The Alfabet Start screen will open the Alfabet interface in another new window and remain in the background.

In most cases, reports that include links to the Alfabet interface are generated based on current data from the Alfabet database, thus combining the links to the Alfabet interface with direct access to the Alfabet database.

> Direct access to the Alfabet database is described in the chapter Accessing the Alfabet Database with External Applications.
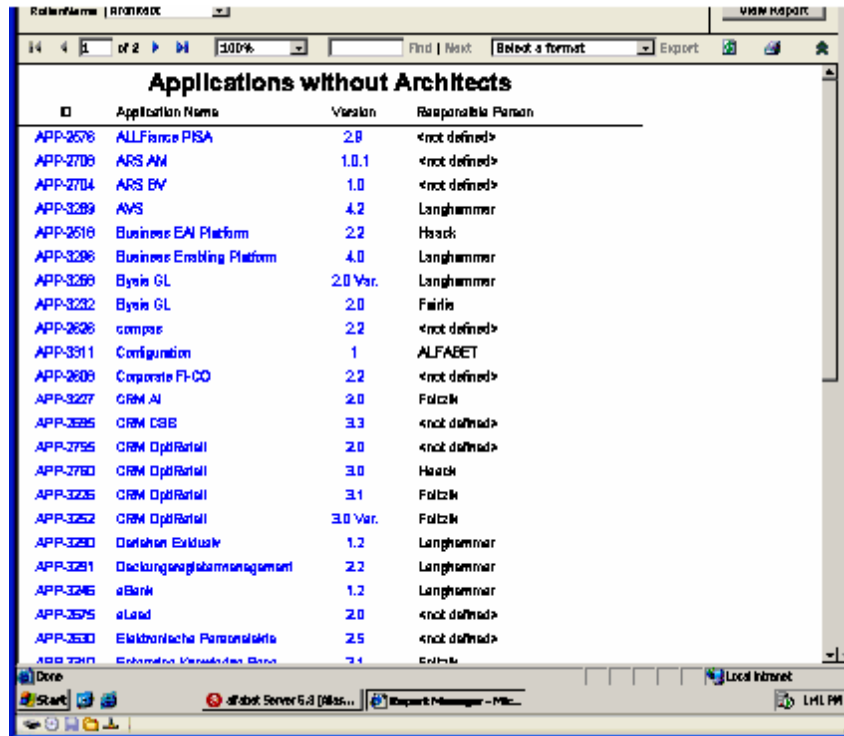
> An IT Manager wants to assign a task to application architects of applications via a workflow in Alfabet. Therefore, the IT manager wants to make sure that the role "Architect" is assigned to a Alfabet user for each application in the department.
>
> The reporting tool is configured to generate a report about all applications in the Alfabet database do not have the role "Architect" assigned. The access is configured as an SQL-based

direct access to the Alfabet database (as described in Accessing the Alfabet Database with External Applications).

For each application displayed in the report, a link is generated by the reporting tool to the Alfabet database. The links are based on the Alfabet specific link syntax for links to the Alfabet database. The link leads to the **Responsibilities** page view of the respective application with Read/Write access permissions for the IT Manager. When the IT manager clicks the link for an application, he accesses the **Responsibilities** page view in Alfabet where he can assign the role "Architect" to the application.

FIGURE: Report showing all applications that do not have the role "Architect" assigned

Link Syntax for Links to Alfabet Views from External Applications

To link to a Alfabet view from an external application, you must use the standard application URL for Alfabet that is also used to access Alfabet from Web clients. The link must end with `ExternalAccess.aspx` followed by a question mark and parameters for opening the view defined as parametername=parametervalue. Parameters are separated with an `&` sign.

⚠️ Please note that URL encoding is required in the link. This means that whitespaces must be written as `%20` and a dot must be written as `%2E`, for example.

A typical link for opening a view for an object would be written in the following syntax:

```
?AccessType=ExternalAccess&UserType=[Named|Anonymous]&Profile=[NameOfUserProfile]&Object=[RefStrOfObject]&View=[ViewType]:[NameOfView]&CID=[LanguageID]
```

with

Parameter	Mandatory	Content
AcessType	Yes	This parameter specifies the type of access to the Alfabet database. It must be set to ExternalAcess.
UserType	No	Access permissions are also specified in the server alias configuration of the Alfabet Web Application with the **Allow Anonymous Login** attribute. Anonymous access is only possible if it is allowed both in the alias configuration and in the external link. Anonymous access is allowed per default in the alias configuration. This parameter is only valid for the standard login. Possible values are: • Named. Restricts access to the view to specified Alfabet users of the type Named User. Named users who can edit the view have either Read/Write or ReadOnly access permissions. • Anonymous: Allow all users of the external application to access the view. Anonymous users have ReadOnly access permissions.
Profile	No	Name of the user profile used for displaying the view. If no user profile is specified, the default user profile for anonymous users is used. **NOTE:** The presentation of data may change depending on the use profile used for log on. For example, it is possible to hide page views that are available by default for an object.
Object	No	The REFSTR property of the object for which the view should be opened. **NOTE**: The REFSTR property is not displayed in the Alfabet user interface. You can read the REFSTR of objects from the Alfabet database for example by creating a configured report returning the REFSTR of objects. **NOTE**: The definition of the object for which the view should be opened is required for all object profiles, most of the Alfabet standard page views and may be required for configured reports. There are two methods for defining the base object. You can either define the object by REFSTR with this parameter or use the parameters BaseClassName and Key_ described below to identify the object by specification of the object class the object belongs to and key object class property values that are unique for the object.
BaseClassName	No	The name of the object class the object for which the view should be opened belongs to. The value must equal the value of the **Name** attribute of the object class. **NOTE**: This parameter is only valid in combination with at least one parameter Key_. **NOTE**: The definition of the object for which the view should be opened is required for all object profiles, most of the Alfabet standard page views and may be required for configured reports. There are two methods for defining the base object. You can either define the object by REFSTR with the parameter Object described above or use the parameters BaseClassName and Key_ to identify the object by specification

Parame- ter	Man- da- tory	Content
		of the object class the object belongs to and key object class property values that are unique for the object.

Key_	No	The name and value of a key property that unambiguously identifies the object for which the view should be opened. The syntax for this parameter is:
		
		`Key_PropertyName:PropertyValue`
		
		You can define multiple parameters `Key_` within the same link if the object cannot be identified unabiguously via a single property. It is recommended that you use the properties defined in the unique class key for the object class.
		
		NOTE: This parameter is only valid in combination with the parameter `BaseClass-Name`.
		
		For example, if the view should open for an application with the name and version "Customer Manager 2.0". The link must contain the following attributes to define the application:
		
		`BaseClassName=Application&Key_Name=Custom%20Manager&Key_Version=2%2E0`
		
		NOTE: The definition of the object for which the view should be opened is required for all object profiles, most of the Alfabet standard page views and may be required for configured reports. There are two methods for defining the base object. You can either define the object by `REFSTR` with the parameter `Object` described above or use the parameters `BaseClassName` and `Key_` to identify the object by specification of the object class the object belongs to and key object class property values that are unique for the object.

View	No	The technical name and type of the view to be displayed. The view must be specified as <ViewType>:<ViewName>. Allowed view types are `ObjectView`, `GraphicView`, `Report`, `BusinessFunction` and `Explorer`. Object views, graphic views displaying data for a defined object and reports with a base object must be available for the object specified with the parameter `Object` or the parameters `BaseClassName` and `Key_`. If an object cockpit shall open, the view must be defined as: `ObjectView:<ObjectViewName>:<ObjectCockpitName>` If the parameter is missing, the object's default object profile is displayed. Explorers can optionally be opened at a defined object node. The hirarchy of nodes in the explorer to open must then be added to the view with the optional parameter `RefPath`.
Con- text- Key_	No	This parameter is only relevant for links opening a configured report to define a filter field name and value pair for pre-setting of filters in a configured report. The syntax for this parameter is:

Parameter	Mandatory	Content
		``
ContextKey_FilterNameWithPrefix:FilterValue
``` You can define multiple `ContextKey_` parameters ithin the same link to pre-set multiple filters. For example, if a configured report based on native SQL contains a filter named @AppName that allows specification of an application name, and the filter shall be preset to "Customer Manager". The link must contain the following attribute to set this filter field: ```
ContextKey_@AppName=Custom%20Manager
``
RefPath	No	This parameter is only relevant for links opening an explorer to open the explorer specified node. The `REFSTR` of all objects in the hierarchy that shall open must be specified in the order of the hierarchy and separated with a vertical bar:. ```
RefPath=<REFSTR of object>|<REFSTR of object>|<REFSTR of
object>
``
CID	No	The ISO identifier of the language to be used to display the view. Possible values are: **1033** for English **1031** for German

In the previous example, the reporting tool searches for the relevant applications in the Alfabet database and reads the *RefStr* property from the database. The link is then generated for each application using, for example, the following basic dynamic link, whereby `{RefStr}` is replaced by the `RefStr` of the respective application:

```
http://company.intern.com/Alfabet/ExternalAccess.aspx?AccessType=Extern
alAccess&UserType=Named&Profile=ApplicationManager&Object={RefStr}&View
=GraphicView:ObjectObligations&CID=1033
```

The user type is `Named` in order to allow the IT Manager to edit the application. The **Responsibilities** page view (`ObjectObligations`) is displayed as shown in the user profile `ApplicationManager` in the English language.

Managing Access Permissions for Access from External Applications

The access permissions for users accessing Alfabet from a link in an external report are determined by the parameter *UserType* specified in the link and the **Allow Anonymous Login** parameter in the server alias configuration of the Alfabet Web Application. Permission also depends on the authentication method used in the company.

Access Permissions for Named Users

If the `UserType` attribute in the link is defined as `Named`, only Alfabet users of the user type `NamedUser` will be able to access the view. The user will have Read/Write access permissions to the view.

> ⚠ Access permissions also depend on the user profile specified in the link. Please keep the following in mind regarding the specification of a user profile in the link:
>
> - If the user profile grants ReadOnly access permissions, even a user of the user type `NamedUser` will not be able to edit the view when logged in with this user profile.
>
> - If no user profile is specified in the link, the default user profile for anonymous users will be used. The default user profile is Read/Only.

- If enterprise authentication is used to authenticate the user, authentication is done automatically and the Alfabet view is directly displayed to the user. If the user is not configured in the Alfabet database or is of the user type Anonymous, an error message will be displayed stating that the login will be denied to the anonymous user.

- If the standard login is used, a login screen will be displayed once the user clicks the link. Authentication with the user name and password will be required.

Access Permissions for Anonymous Users

If the user type in the link is defined as anonymous, access permissions will depend on the XML attribute `AllowAnonymousUser` in the `AlfabetMS.xml` configuration file of the Alfabet Web Application.

The XML attribute `AllowAnonymousLogin` is activated by default and any person clicking the link can access the Alfabet view without being required to login. The user will have ReadOnly access permissions to the view.

If the parameter `XML attribute` is deactivated, the accessibility of the view will be different for standard login and enterprise authentication.

If enterprise authentication is used to authenticate the user, authentication is done automatically and the Alfabet view will be displayed to the user if the user is configured in the Alfabet database with the user type NamedUser. Otherwise, an error message will be displayed stating that the login is denied to the anonymous user.

If standard login is used, access to Alfabet views will be denied for all users and an error message will be displayed stating that the login is denied to the anonymous user.

You can edit the XML attribute `AllowAnonymousLogin` in the tool Alfabet Administrator.

To edit access permissions for anonymous users:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer and click the server alias of the Alfabet Web Application that you want to configure access permissions for.

2) In the toolbar, click the **Edit** button. An editor opens.

3) In the **Server Settings** > **Security** tab, select the **Allow Anonymous User** checkbox to allow anonymous users to access Alfabet views from external applications. If you deselect the checkbox, access will be denied for all users except named users that are authenticated via enterprise authentication.

Deactivating Access from External Applications

By default, it is possible to link to Alfabet views from external applications. The `AlfabetMS.xml` configuration file contains a parameter that allows you to prevent any access to Alfabet from an external application. The `AlfabetMS.xml` configuration file is edited in the tool Alfabet Administrator.

To deactivate access to Alfabet from external applications:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer and click the server alias of the Alfabet Web Application that you want to configure access permissions for.

2) In the toolbar, click the **Edit** button. An editor opens.

3) In the **Server Settings** > **Security** tab, select the **Not Allowed** option in the **External Access** field in order to prevent access from external applications.

Integration of External Reports into Alfabet

You can make external reports available to users via the Alfabet interface. The external report is independent of Alfabet and is therefore addressed by the Alfabet Web Application by means of a link to a URL. When a user activates the link, the report opens in a new Web browser window.

An external report can have any type of content. For example, the external report could contain static text or a script. It can present data from external sources or data derived from the Alfabet database by direct SQL-based access to the Alfabet database (see Accessing the Alfabet Database with External Applications).
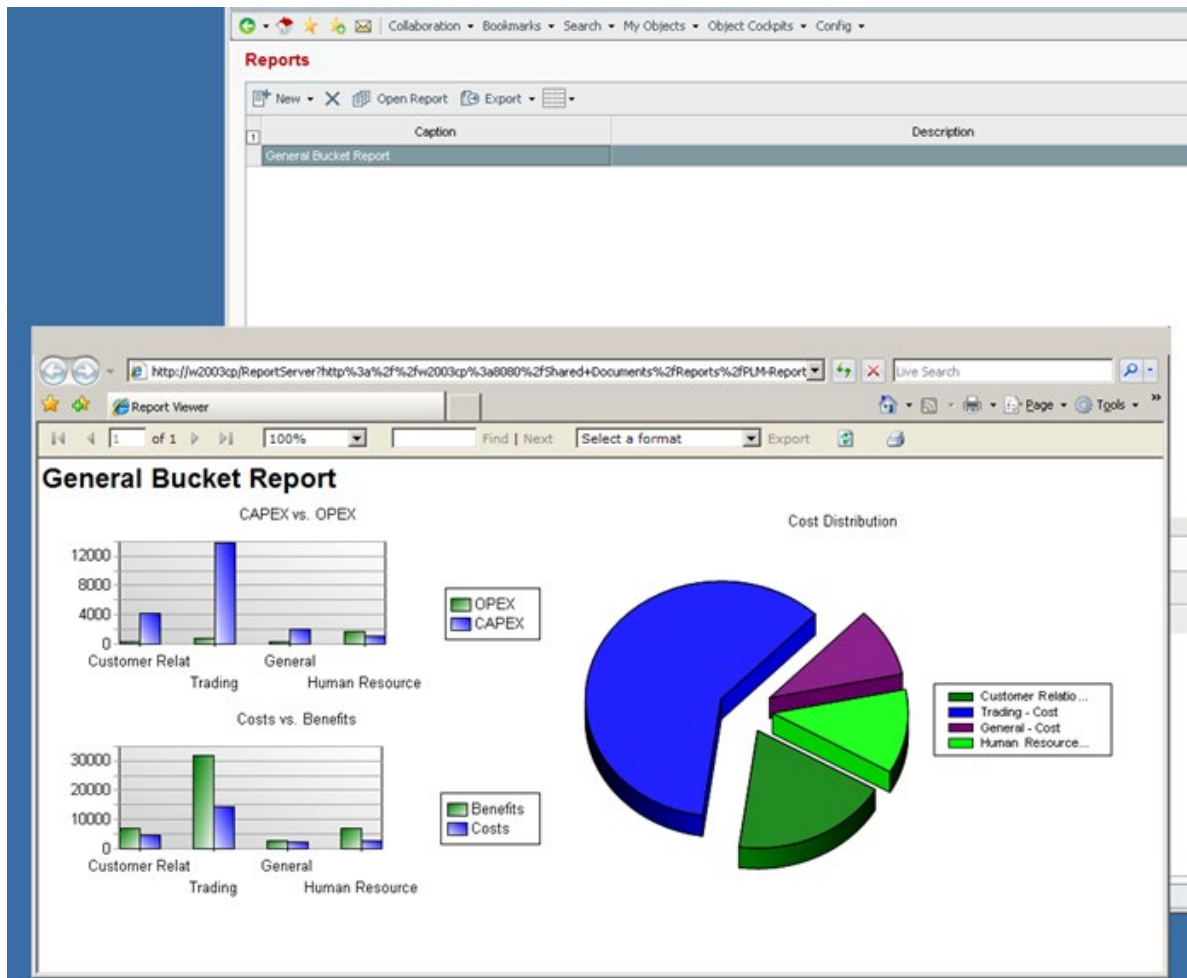
FIGURE: Accessing an external report from the Alfabet interface using the button Open Report

In the figure above, the external report is a Web application that is located parallel to the Alfabet Web Application on the same Web server.

Access to the external report must be configured by an administrator responsible for the solution configuration of Alfabet. Alfabet users can then view the external report in the **Configured Reports** functionality (SRCH_Reports_Only). The **Configured Reports** functionality displays all configured external reports with a short description (optional) of the report. To open the external report, the user must select the report in the table and click the **Open Report** button. The report opens in a new window.

The following steps are necessary to allow Alfabet users to access an external report via the Alfabet interface:

- A report must be configured in the Alfabet configuration tool Alfabet Expand. The report must specify the following:

  - The name of the external report that will be displayed in the Alfabet interface.

  - A comment about the external report that will be displayed in the Alfabet interface.

  - The URL of the external report.

  - Specification defining whether the report will be available via the **Configured Reports** functionality (SRCH_Reports_Only) or via the **Configured Reports** page

view (`ObjectReportsDataSet`) in the object profiles of all objects of one or multiple specified object classes.

- The status of the external report. New external reports are created with the state Plan. As long as the state is Plan, the configuration can be edited and the report will not be displayed in the Alfabet interface.

  For information about the configuration of an external report, see the section *Creating an Alfabet Configured Report That Opens an External Report* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- The report state of the external report must be set to Active. It will then be visible to authorized users in the **Report Administration** functionality available via an administrative user profile and the **Configured Reports** functionality.

- The user administrator can assign access permissions for the report to Alfabet users in the **Report Administration** functionality available via an administrative user profile. For more information about defining user access to external reports, see the section *Defining and Managing User Access to Configured Reports* in the reference manual *User and Solution Administration*.

- Once the steps listed above have been completed, the user will be able to access the reportvia the **Configured Reports** functionality (`SRCH_Reports_Only`) or via the **Configured Reports** page view (`ObjectReportsDataSet`) available in the object profile of the relevant objects.

SOAP-Based Third-Party Access to the Alfabet Database via Web Services

Software AG provides Web Services that are designed to allow direct access to the data stored in the Alfabet database from external programs that can run on different platforms. Platform interoperability is based on SOAP protocol and offers the possibility to make remote procedure calls via HTTP to the Alfabet data exchange service from programs that are written using different programming languages and that run on different operating systems.

Alfabet provides the Web Service *AlfaQueryWebService* to read data from Alfabet database.

Calls to the methods of the service can be made from other programs or HTML forms using SOAP or POST format. The Web Service connects to the running Alfabet application server and return XML documents as a result. The calling program must handle exceptions that can occur in service to control the success of the method call.

WSDL descriptions for the Web Service is available from the standard interface *AlfaQueryWeb-Service.asmx?wsdl*

For a detailed description of the Web Services and their operation, see the reference manual *Web Services for Alfabet*.

Integrating Data from External Sources

Software AG provides an interface for the one-way integration of objects from external sources (LDAP or database tables). This includes update of object data in the Alfabet database with data from the external source as well as redirection of selectors in the Alfabet user interface to the data in the external data source. If user data is integrated from an external source, the data from the external source can additionally be used to identify the user during login to the Alfabet user interface.

> For example, a company maintains the contact information about its employees in LDAP. The directory is updated regularly. The company does not want to maintain two parallel employee databases and therefore wants the user data in the Alfabet database to be synchronized with the updated LDAP instead of being edited manually by a responsible user administrator in Alfabet. The company therefore uses the Alfabet interface for integration of object data to display LDAP data in the Alfabet user interface and perform login on basis of the LDAP data. Additionally, LDAP data is imported to the Alfabet database in regular intervals to maintain an updated Alfabet internal user database. The Alfabet internal user data is required by Alfabet internal mechanisms such as the evaluation of mandate-related access permissions.

The interface is available for the following source types:

- LDAP and LDAPS (in the following the name LDAP is used to refer to both LDAP and LDAPS.)

- Microsoft® SQL Server® databases

- Oracle® Server databases

- Microsoft® Access® databases

Data integration can be done from one external source or from a pool of different external sources.

The Alfabet interface can be configured by the customer to specify which sources are used for data import and to map Alfabet classes and properties to the external source data. Only the class properties that are included in the mapping information are synchronized with the external source. The configuration required to synchronize data with an external source is described in the section Configuring the Implementation of External Sources for Data Synchronization.

Mechanisms for Data Integration

The interface for external source synchronization includes three mechanisms for integration of data from external sources:

- On a per-object basis whenever an object is selected in Alfabet.

    The Alfabet user interface can be configured to display data from the external source instead of data from the Alfabet database in selectors. For example, if user data from an LDAP source is imported, then each time you assign a user as authorized user to an object, you select the user from the LDAP source via the Alfabet selector. The Alfabet database is not updated as long as the external object can be mapped to an existing object in the Alfabet database. Only in case the object from the external database cannot be mapped with an object in the Alfabet database, the new object is added to the Alfabet database. For more information, see the section Synchronizing Data by Accessing an Object.

- On a per-user basis whenever a user logs in to the Alfabet user interface and login is performed via the data from the external source.

   The Alfabet Web Application can be configured to perform user authentication on basis of data from the external data source instead of using data from the Alfabet database. When a user logs in to the Alfabet user interface, data of this specific user is updated in the Alfabet database. For more information, see the section Configuring the Alfabet Web Application to Perform User Authentication Based On User Data from an External LDAP Data Source.

- Via a batch job process that synchronizes all mapped objects in the Alfabet database with the external source.

   The batch job cannot be started from the Alfabet interface but must be started via a command line or update script initialized by the responsible system administrator. For more information on running the batch job, see the section Synchronizing Data via a Batch Process.

   ⚠ The batch job process only updates existing Alfabet objects. New objects are not created in the Alfabet database. If a new object is specified in the external data source, it will not be included in the Alfabet database by means of the batch job.

⚠ Synchronization overwrites the data in the Alfabet database and manual changes made in Alfabet are lost the next time synchronization with the external source is performed.

It is possible to synchronize data from a single external source or from multiple external sources. When multiple external sources are used, the configuration of the interface for external source synchronization allows either a source pool to be configured or all sources to be defined independently of each other:

- **Configuring multiple independent external sources**: Multiple external sources can be configured without implementing an external source pool. In this case, each external source should be mapped to a different Alfabet object class. For the batch synchronization of data, an external source from the external source configuration can be selected in the command line.

- **Configuring external source pools**: Multiple external sources can be defined in an external source pool and the source pool is then mapped to an object class. The user can select an external source from the external source pool in a drop-down field available in the object selector. The list displays the captions of all external sources in the external source pool. The tool for batch synchronization of data can be configured to use all external sources in the external source pool or only one external source in the external source pool. It is possible to specify more than one external source pool. Batch synchronization can be executed with any of the configured external source pools or with a single source in the external source pool.

Synchronizing Data by Accessing an Object

⚠ Synchronizing data with external sources affects the search for objects in Alfabet:

- It is not possible to use the operator `BETWEEN` when searching for users that are synchronized with user data stored in an external LDAP source.

- Wildcards are not added automatically to search strings.

- Whether the search is case sensitive or case insensitive depends on the settings in the external source configuration. For more information, see Configuring the Implementation of External Sources for Data Synchronization

When a user searches for an object in Alfabet that is synchronized with an external source, the data stored in the external source is displayed in the Alfabet selector rather than the data in the Alfabet database. The external source may contain data that differs from the content of the Alfabet database.

The search for and selection of an object has no effect on the data in the Alfabet database if the object is available in both the external source and the Alfabet database. Only if the object does not exist in the Alfabet database, a new object is created in the Alfabet database. Values defined for all mapped properties are written to the Alfabet database. The following behavior applies:

Object Available in External Source	Object Available in Alfabet	Influence on Access of Object
Yes	Yes	Data in the Alfabet database is not altered.
Yes	No	A new object is created in the Alfabet database. Values defined for all mapped properties are written to the Alfabet database
No	Yes	The object is not displayed in the selector and therefore selection is not possible.

⚠ With the exception of object selectors and other search functionalities, only objects stored in the Alfabet database are displayed in the Alfabet interface. For example, if a user administrator accesses the **Users Administration** functionality, the data in the Alfabet database for the class Person will be displayed in relevant views even if the class Person is mapped to an external source. It is possible that the administrator could manually edit a property for an object of the class Person without knowing that the property is mapped to an external data source. The values entered by the administrator will be overwritten the next time the object is synchronized with the external source (for example, via a batch job). Only changes made to properties that are not mapped to the external source will be persistent.

Therefore, it is crucial that the relevant Alfabet users responsible for any objects mapped to an external source are informed in detail about which properties are mapped to the external source and therefore should NOT be edited in the Alfabet database.

ⓘ The following applies if the class Person is mapped to user data in an external table or LDAP:

- For the object class Person data can be maintained in parallel in the external source and the Alfabet database. Person selectors can show both data from the external source(s) and from the Alfabet database in the same data set. If **Show Internal Persons** is selected, the data from the Alfabet database is displayed in parallel to the data of the external source. The background color of the data set informs the user about the availability of the data in the sources:

- Green: The person exists in both the Alfabet database and the external source

- Yellow: the person exists in the external source only

- No color: the person exists in the Alfabet database only

For example, this mechanism is helpful to configure temporary users such as contractors that shall access Alfabet for a limited time that do not need to be added to the external source that is used to maintain employee data.

- On the **Users** page view for user groups, the **New** button will display a sub-menu that allows the user to decide whether to add a person existing in the Alfabet database or a person defined in the external source (option **Add Person from External Source**). If a person from an external source is added, the data for that person will be updated in the Alfabet. If the user does not exist in the Alfabet database, a new user will be added to the Alfabet database.

- It is not possible to use the operator `BETWEEN` when searching for objects that are synchronized with an external source pool.

- Wildcards are not automatically added to search strings.

- Whether the search is case sensitive or case-insensitive depends on the settings in the external source configuration. For more information, see Configuring Access to an External Data Source and Mapping of Data.

- Search strings and attribute values for LDAP objects can contain the following special characters: & | > < ~ / \

The class Person, which specifies user information in Alfabet, is synchronized with an LDAP source. The following examples illustrate various scenarios that may be affected by data synchronization:

- **Data for a new employee was added to the LDAP source. The employee will use Alfabet with the user profile Application Architect. The user administrator must define access permissions and user information for the new user.**

  In the Alfabet interface, the user administrator opens the **User Profiles Administration** functionality, navigates to the object profile of the user profile Application Architect and opens the **Users** page view in the **Basic Data** workspace. In the toolbar, the user administrator clicks **New** > **Add Person from External Source**. A selector opens that displays the data from the external LDAP source. The administrator selects a user and clicks **OK**. The user is displayed in the table as a user assigned to the user profile Application Architect. At the same time, the new user has also been created in the Alfabet database. All mapped properties are automatically updated for the new user in Alfabet. The user administrator can now switch to the **Users Administration** functionality to edit the user data such as password specification, etc. for the new user.

- **The user administrator wants to change the email address and the technical name of a user.**

  In the Alfabet interface, the user administrator opens the **Users Administration** functionality. The data displayed in the table is the data from the Alfabet database. The administrator selects the user in the table and clicks the **Edit** button. The **User** editor opens.

  The email address is a property that is configure to be mapped to the external source. The user administrator could edit the user's email address in Alfabet, but the email

address would be overwritten with the next data synchronization. However, the user administrator has been informed about the mapping of the relevant property to the external source. Therefore, the user administrator changes the email address in the external LDAP source only. The next time the user information is synchronized with the Alfabet database, the email address located in the external source will be updated to Alfabet.

In contrast, the property for the technical name of the user is not configured to be mapped to the external source. Therefore, the user administrator edits the property directly in Alfabet database and saves the changes.

Synchronizing Data with an External Source Pool (per Object Synchronization)

If synchronization is carried out from multiple external sources, the Alfabet user must select the external source in the source pool that data is to be synchronized with. The user selects the external source in the relevant object selector by opening a drop-down field displaying the available external sources in the external source pool. Then the user can search and select the relevant object in the selected external source. The subsequent steps for synchronization are then carried out as described above for synchronization with a single data source.

In the example below, the user can select between two LDAP sources to search for a user. After selecting the external source in the drop-down field, the user must click the **Search** button to display the data from the selected external source in the results table.
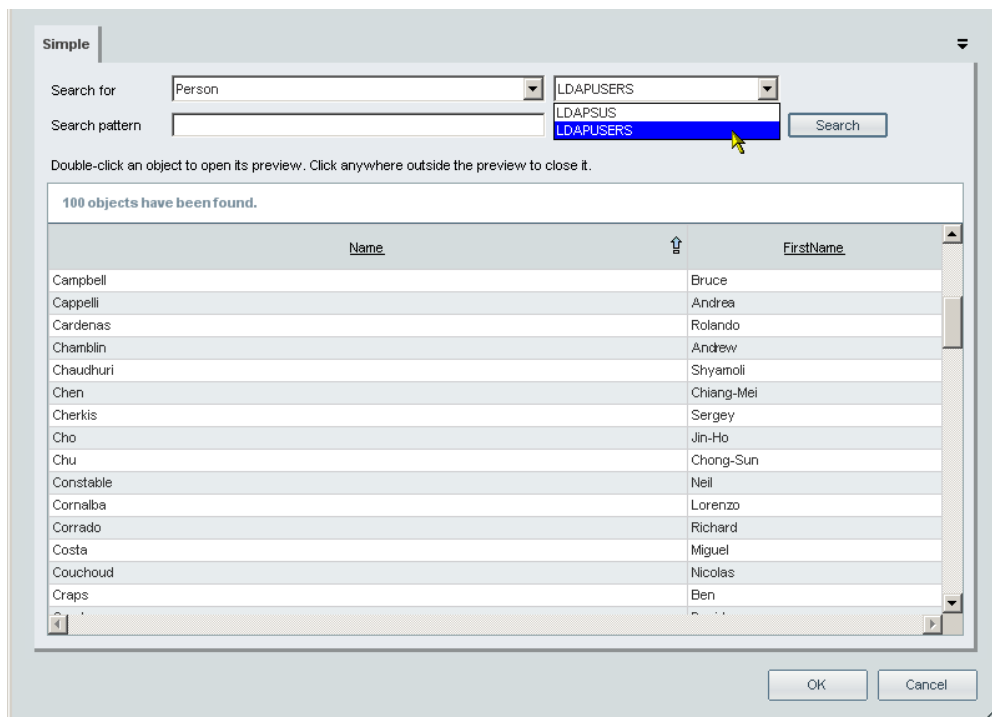
FIGURE: Drop-down list to select between multiple external sources in a source pool

Performing User Authentication Based on User Data from an External Data Source

If the object class `Person` is mapped to data of an external data source, the user data from the external source can be used to identify and authenticate the user during login to the Alfabet user interface. When a user logs in, the data about the user in the Alfabet database is updated with the mapped data from the external data source.

Synchronizing Data via a Batch Process

The import and one-way synchronization of objects in the Alfabet database with data from an external data source (pool) is carried out via a batch process using the executable `ExternalSourceSynchronization.exe` provided by Software AG.

The action performed during batch synchronization depends on the availability of the object in both the Alfabet database and the external database. The following behavior applies:

Object Available in External Source	Object Available in Alfabet	Action During Batch Synchronization
Yes	Yes	Vales defined for all mapped properties are updated in the Alfabet database.
Yes	No	No action.
No	Yes	The object in the Alfabet database will not be deleted because dependent objects may exist. However, a property can be configured to indicate that the object in the Alfabet database is marked for deletion.
		The property marking an object for deletion must be configured for the relevant object class in the configuration tool Alfabet Expand:
		• Specify a custom property of the data type "Boolean" for the relevant object class in Alfabet. A flag attribute with the name "`DeletionRequested`" is preconfigured for the class `Person`. For more information about the configuration of custom properties, see the section *Configuring Custom Properties for Protected or Public Object Classes* in the reference manual *Configuring Alfabet with Alfabet Expand*.
		• In the ***ExternalSourceConfiguration*** XML object, you must define the custom property in the XML attribute `AlfaDeletedFlagProperty`. The property value will be set to `true` if the object does not exist in the external data source. For more information see [Configuring Access to an External Data Source and Mapping of Data](#).

The class `Person`, which specifies user information in Alfabet, is synchronized with an LDAP source. A Alfabet user has left the company and the data for the user was deleted from the LDAP source. The user must also be deleted from the Alfabet database. This is carried out as described below:

1) The system administrator for Alfabet executes the batch job via the executable `ExternalSourceSynchronization.exe` to synchronize the Alfabet database with the LDAP source. The user that is no longer specified in the LDAP source is marked for deletion in the Alfabet database.

2) In the Alfabet interface, the user administrator opens the **Users Administration** functionality and reviews that a checkmark is set for the user (=`True`) in the **Deletion Requested** column. The user administrator navigates to the user's user profile and assigns all objects that the deleted user was responsible for to other Alfabet users. After all objects have been reassigned, the user administrator deletes the user from the Alfabet database.

Synchronizing Data with an External Source Pool (per Batch Synchronization)

If an external source pool is configured to map the objects in the Alfabet database to multiple external sources, the batch job for data synchronization must be configured for the external source pool. The command line to start the batch tool allows you to specify the source for the batch process. It is possible to select a single external source in the external source pool for the synchronization or to synchronize the data with all external sources in the external source pool. By default, the batch job synchronizes the mapped objects in the Alfabet database with the data of the first external source specified in the external source pool configuration.

If multiple external source pools are defined, the batch job allows you to select the external source pool used for data synchronization as well as to select one specific external source from a selected external source pool.

For more information about configuring external source pools, see the section Configuring the Use of External Source Pools.

If you synchronize the data in the Alfabet database with the data from multiple external sources in one batch process, you must ensure that the external sources do not contain conflicting data. If you cannot ensure integrity of data to be imported, it is recommended that you use the data import mechanisms provided by the Alfabet Data Integration Framework (ADIF) instead of batch synchronization via the `ExternalSourceSynchronization.exe`.

Executing a Batch Job to Import Data from External Sources

The interface for data synchronization with external sources must be configured prior to using the batch tool. For information about the configuration of the interface, see Configuring the Implementation of External Sources for Data Synchronization.

The import of data will be triggered by a batch utility:

Executable:	`ExternalSourceSynchronization.exe` located in the Programs subdirectory of the Alfabet installation directory
Remote or stand-alone access:	Stand-alone access to the database with a server alias or remote access with a remote alias, connecting to a running Alfabet Server.
Precondi-tions:	The interface for data synchronization with external sources must be configured prior to using the batch tool. For information about the configuration of the interface, see Configuring the Implementation of External Sources for Data Synchronization.
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see Standard Logging for Alfabet Batch Utilities.
Command line help	Start executable with-h or-help

The following command line starts the batch job for the synchronization of data with an external data source:

```
ExternalSourceSynchronization.exe-msalias <Alias Name>-alfaLoginName <user
name>-alfaLoginPassword <user login password>-source <name of external
source>-pool <name of the external source pool>
```

The table below displays the command line options:

Command Line Op-tion	Manda-tory/ Op-tional	Explanation
`-msalias` <alias name>	Manda-tory	Enter the alias name as specified in the `AlfabetMS.xml` configura-tion file. If a remote alias is specified, the corresponding Alfabet Server must be running. If a server alias is specified, all other Alfabet applications with access to the Alfabet database must be stopped.
`-msaliasesfile` <Alfabet configuration file path>	Op-tional	If the `AlfabetMS.xml` configuration file that contains the specifica-tion of the alias is not located in the same directory as the executa-ble, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName` < Alfabet user name>	Manda-tory	User name for login to the Alfabet Server. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.

Command Line Option	Mandatory/ Optional	Explanation
`-alfaLoginPassword` < Alfabet login password>	Optional	Password for login to the Alfabet Server.
`-source` <name of external source>	Optional	This parameter is only needed if more than one external data source is specified in the XML object ***ExternalSourceConfiguration*** or if only one external source in an external source pool will be used for synchronization. If an external source in an external source pool is specified, the parameter`-pool` is also required. If the parameter-`source` is not specified and the parameter-`pool` is specified, all external sources in the external source pool are used for synchronization. If neither-`source` nor-`pool` are specified, the first external source specified in this XML object ***ExternalSourceConfiguration*** is used.
`-pool` <name of external source pool>	Optional	Name of the external source pool as specified in the XML object ***ExternalSourceConfiguration***. This parameter is required to synchronize data from an external source pool. If the parameter-`source` is set in combination with the parameter-`pool`, only the specified external source in the external source pool is used for synchronization.
`-disposeafterquery` <true/false>	Optional	The default for this parameter is either a preconfigured value or `false`.
`-idletime`	Optional	The batch utility will wait for alive messages from the Alfabet Server and if none are received within 60 seconds, the job will be cancelled. The allowed wait time for alive messages can be changed by defining a new time interval in seconds via the-`idletime` command line option.

Configuring the Implementation of External Sources for Data Synchronization

⚠ External data access (for example, redirecting object selectors) must be configured for your Alfabet solution by Software AG Support. You will be able to configure your individual interface settings only after the basic configuration of the Alfabet solution has been defined.

The following configuration is required to synchronize the Alfabet database with an external data source:

- If data synchronization is to be performed on a per-object basis, the Alfabet interface must be configured to display data from external sources when a user opens a selector.

This requires the configuration of two XML objects that can be edited using the tool Alfabet Expand:

- An external object selector definition to access the external data source must be created per object class that shall be synchronized with the external source. This can only be performed by Software AG Support. The selector definition in an XML object ***SelectorDef*** is then visible and editable for the customer in the tool Alfabet Expand or optionally in the tool Alfabet Administrator.

- Configure Alfabet to use the external object selector instead of the default object selector in the XML object ***GeneralViewMap***. It is recommended that configuration is done by Software AG Support and that you do not modify the XML object ***GeneralViewMap*** provided by Software AG.

- Specifications regarding access to the external source and the mapping of objects must be configured in the XML object ***ExternalSourceConfiguration***, available via the configuration tool Alfabet Expand or the tool Alfabet Administrator.The tool Alfabet Administrator provides an interface for easy configuration of external sources without directly editing the XML object. For more information, see Configuring Access to an External Data Source and Mapping of Data.

- Definition of the connection properties as well as the object class and property mappings for external data import and synchronization must be configured in the XML object ***ExternalSourceConfiguration***, available via the configuration tool Alfabet Expand or the tool Alfabet Administrator. The tool Alfabet Administrator provides an interface for easy configuration of external sources without directly editing the XML object. For more information, see Configuring Access to an External Data Source and Mapping of Data.

- If external user data is to be used for authentication during the login of users to the Alfabet user interface, the server alias must be configured to use the external source for user authentication. For more information see Configuring the Alfabet Web Application to Perform User Authentication Based On User Data from an External LDAP Data Source.

- If external user data is to be used for authorization configuration, the mapping of external relations to the relations table in the Alfabet database must be defined in the XML object ***AuthConfiguration*** or a copy thereof. For more information, see Configuring the Alfabet Web Application to Perform User Authentication Based On User Data from an External LDAP Data Source.

You can configure multiple external sources or multiple independent external sources or an external source pool:

- **Configuring external sources**: Multiple external sources can be configured without implementing an external source pool. In this case, each external source should be mapped to a different Alfabet object class. For the batch synchronization of data, an external source from the XML specification can be selected. You must add an XML element `External Source` to the XML element `ExternalSourcesConfiguration` for each external source configuration. You can configure the connection parameters for the external source by changing the XML attributes of the XML element `External Source`.

- **Configuring external source pools**: The user can select an external source from the external source pool in a drop-down field available in the object selector. The list displays the captions of all external sources in the external source pool. The tool for batch synchronization of data can be

configured to use all external sources in the external source pool or only one external source in the external source pool. You must add an XML element `ExternalSourcePool` to the XML element `ExternalSourcesConfiguration` for each external source pool configuration. For each external source in the external source pool, you must add one or more child XML elements `External Source` to the XML element `ExternalSourcePool`. You can configure the connection parameters for the external source by changing the XML attributes of the XML element `External Source`. It is possible to specify more than one external source pool. Batch synchronization can be executed with any of the configured external source pools.

The following information is available:

- [Configuring the Alfabet User Interface for Data Synchronization with External Sources](#)

- [Configuring Access to an External Data Source and Mapping of Data](#)

- [Configuring the Use of External Source Pools](#)

- [Configuring the Alfabet Web Application to Perform User Authentication Based On User Data from an External LDAP Data Source](#)

  - [Configuring Authentication Based On Login with One of Multiple Permissable LDAP IDs](#)

- [Configuring the Alfabet Web Application to Perform User Authorization Based On User, User Group and User Profile Data from an External Data Source](#)

  - [Configuring the XML Object ExternalSourceConfiguration to Support User Authorization](#)

  - [Configuring the Update of Alfabet Internal Relations in the XML Object AuthConfiguration](#)

  - [Activating User Authorization in the Remote Alias of the Alfabet Client](#)

Configuring the Alfabet User Interface for Data Synchronization with External Sources

If the data of a specific object class in the Alfabet database is synchronized with data of an external source, the object selectors in the Alfabet interface will display the data from the external source instead of the data from the Alfabet database.

The following must be carried out by Software AG Support in the tool Alfabet Expand in order to configure the redirection of object selectors:

- Create a selector definition to access the external data source. This requires the configuration of an XML object **SelectorDef**.

- Configure Alfabet to use the external object selector instead of the default object selector in the XML object **GeneralViewMap**.

The selector definitions in XML objects **SelectorDef** cannot be created by the customer, but the XML objects provided by Software AG are visible and editable in Alfabet Expand. Nevertheless it is recommended that you NOT edit selector definitions or the XML object **GeneralViewMap** for external sources without support from Software AG Support.

By default, the XML objects are visible in Alfabet Expand only. You can make any XML object displayed in Alfabet Expand visible in the Alfabet Administrator by setting the **Visible in Administrator** attribute of the XML object to `True` in Alfabet Expand.

To view the XML objects **SelectorDef** defined by Software AG Support for your external source configuration:

1) In Alfabet Expand, go to the **Presentation** tab, expand the **XML Objects** node, and then expand the **ObjectSelectors** folder.

2) Double-click the relevant selector definition. The XML object configuration opens in the center pane. The following table displays the XML elements and XML attributes of the XML object **SelectorDef** in order to help you to understand the predefined configuration:

XML Element (Bold) / XML Attribute	Description
ObjectSelectorDef	
Name	Specifies the name of the selector definition. This name is used in the XML object **SelectorDef** to identify the object selector.
Pages	For selector definitions in the scope of external source synchronization, the XML attribute `Pages` must be set to `"SimplePage"`. `"Browse"` and `"FullTextSearch"` are not available for selection of data from an external source.
ExternalSource	Specifies the external data source from which data is provided in the selector. The external source is defined by the name of the external source configuration in the XML object **ExternalSourceConfiguration** that defines access to and data mapping for the external source.
Class	
Name	Specifies the object class the selector definition is valid for. The class is defined by the **Name** attribute of the object class in the Alfabet meta-model.

To view the XML object **GeneralViewMap** configured by Software AG Support for your external source configuration:

1) In Alfabet Expand, go to the **Presentation** tab.

2) Expand the **XML Objects** node.

3) Double-click the **GeneralViewMap** node. The XML object configuration opens in the center pane.

The following table lists the XML elements and attributes of the XML object **GeneralViewMap** to help you understand the predefined configuration:

Element (Bold) / Attribute	Description
AlfaViewMap	
`Name`	Defines the name of the XML object.
MapEntry	
`Type`	Defines the type of interface element configured in the XML element ***MapEntry***. The XML attribute `Type` is predefined as `"Selector"` in order to redirect data selection to external sources.
`Source`	Defines the selector that shall be substituted by the selector defined for external source synchronization.
`Target`	Defines the selector defined for external source synchronization provided by Software AG Support.

Configuring Access to an External Data Source and Mapping of Data

To configure access and data mapping for a single external data source:

> (i) Data sources can alternatively be configured in a source pool. The configuration of a single source in a source pool is identical to the configuration described below. The configuration of the parent source pool is described in the section Configuring the Use of External Source Pools.

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer and right-click the server alias of the Alfabet Web Application for which you want to configure the interface for data import and select **Connect**.

2) In the explorer, click the **External Sources Configuration** node and select one of the following:

   • **Add New LDAP External Source** to configure synchronization with an LDAP table.

   • **Add New Oracle External Source** to configure synchronization with an Oracle® database.

   • **Add New SQL Server External Source** to configure synchronization with a Microsoft® SQL Server database.

   • **Add New Access External Source** to configure synchronization with a Microsoft® Access database.

   A new external source node is added as a subordinate node to the **External Sources Configuration** node.

3) Click the new external source node. The attribute window opens in the right pane.

4) Edit the following attributes if applicable:

- **Active**: Set to `True`, to allow connections to the external source to be established. Set to `False` if you want to disable the external source (for example, because the configuration based on the external source is currently reconfigured).

  > ⓘ When authentication is based on an external source and the **Active** attribute of the external source is set to `False`, an error message is displayed to the user when he/she tries to log in.

- **Assembly Class**: This field is used only if Software AG has provided a customer-specific `DLL` file for login to the external source. The assembly class must be specified in this field.

- **Assembly Name**: This field is used only if Software AG has provided a customer-specific `DLL` file for login to the external source. The name of the `DLL` must be specified in this field.

- **Caption**: Define a caption for the external source that will be displayed in the Alfabet interface, for example, in drop-down fields. This attribute is mandatory for external sources that are configured within an external source pool.

- **Ignore Auto Wildcard**: When the Alfabet Web Application is configured to automatically add wildcards to the beginning and end of search strings entered by users in the Alfabet user interface, the automatic wildcard setting will be applied to all searches in selectors pointing to the external database. This may lead to performance issues caused by a non-indexed search of the external data source. Setting the **Ignore Auto Wild Card** attribute to `True` in the external source configuration will suppress the wildcards from being automatically set in selectors that point to the external database. This will help to prevent performance issues.

- **Name**: Define a unique name for the external source. The name is used to identify the external source in other parts of the external source configuration (like, for example the configuration of selectors).

- **Records Max. Count**: Enter the maximum number of records in the range of 0- 1000 that can be read from the source.

  > ⓘ When the number of records found in the external data source exceeds the setting of the **Records Max. Count** attribute, selectors will display a message for the dataset explaining that only a subset of data is available via the search results due to the restriction of the number of data records. The user can refine the search criteria in order to reduce the number of search results.

5) Right-click the external source node and select **Add New External Connection**. A new subordinate connection node is added to the external source node and the attribute window displays the attributes of the subordinate connection node.

6) Define the attributes required to provide the conditions to establish a connection to your external data source:

  > ⓘ Server variables can be used in the connection string, user name and password. Server variables allow you to define all or part of the definition in the server alias configuration of the Alfabet Web Application instead of directly defining it in the external source configuration. Use of server variables is useful, for example, when using the configuration in a test and production environment with different external sources. The same external source definition can be used in both environments. The server alias definition used in

the test and production environment define the correct connection data for the external source used in the respective environment.

- **Authentication Mode**: If the external source is an LDAP server, enter one of the following authentication modes:

  - `Anonymous` for an anonymous connection without providing user name and password.

  - `ServerBind` or `null` for a connection with a check for correct user name and password.

  - `Secure` or `SecureSocketsLayer` for a secured connection.

- **Case Sensitive**: Select `True` if the search for objects in the external database should be case-sensitive. Select `False` if the search for objects in the external database should not be case-sensitive.

  > ⚠️ This setting is ignored for synchronization with LDAP tables. Searches on LDAP are always case-insensitive.

- **Connection String**: Enter the connection string used to connect to the external source (containing source location, user name, and other parameters, according to the requirements of the source). It is recommended that you define the connection string as server variable in the server alias configuration of the Alfabet Web Application and reference the variable with $<ServerVariableName> in the connection string.

- **Dispose After Query**: Select `True` to close the connection between the Alfabet Web Application and the external database server after each request and reopen it for the next request. If you select `False`, the Alfabet Web Application will establish a constant connection to the external database server that is used to handle all requests.

- **Login Name**: Enter the login name for login to the external source. The login name is displayed as asterisks (*) instead of the typed in name and stored encrypted for security reasons.

- **Login Password**: Enter the login password for login to the external source. The login password is displayed as asterisks (*) instead of the typed in password and stored encrypted for security reasons.

- **Use Sort Options**: Select `True` to sort search results in ascending alpha-numeric search order on the external source before transmitting data to the Alfabet Web Application. Select `False` to transmit data without prior sorting of result data sets.

  > ℹ️ Note the following about sorting:
  >
  > - Sorting is performed on the external database server. Activation of sorting should be considered carefully because it can lead to performance problems.
  >
  > - Sorting can be performed when synchronizing with an LDAP source but cannot be guaranteed for all external data sources.
  >
  > - The number of records that can be processed by the Alfabet Web Application is configurable and can be set to any number of records between 1 and 1000. If the number of records in the external source exceeds the configured maximum number of records processed by the

Alfabet Web Application, sorting will influence the selection of data available to the Alfabet user.

If you define synchronization with an LDAP table, a Microsoft SQL Server or Oracle database, you can map data in an interface that is easier to use. In the configuration interface, you can connect to the external database to test your connection configuration and to view the structure of the external database. Right-click the external source node and select **Configure External Source**. The configurations described in the next steps of this procedure can then be performed in the configuration interface.

When you open the configuration interface the external source configuration you can see your already configured external source node in the upper middle pane. When you click the node, the attributes are displayed in the upper right pane.
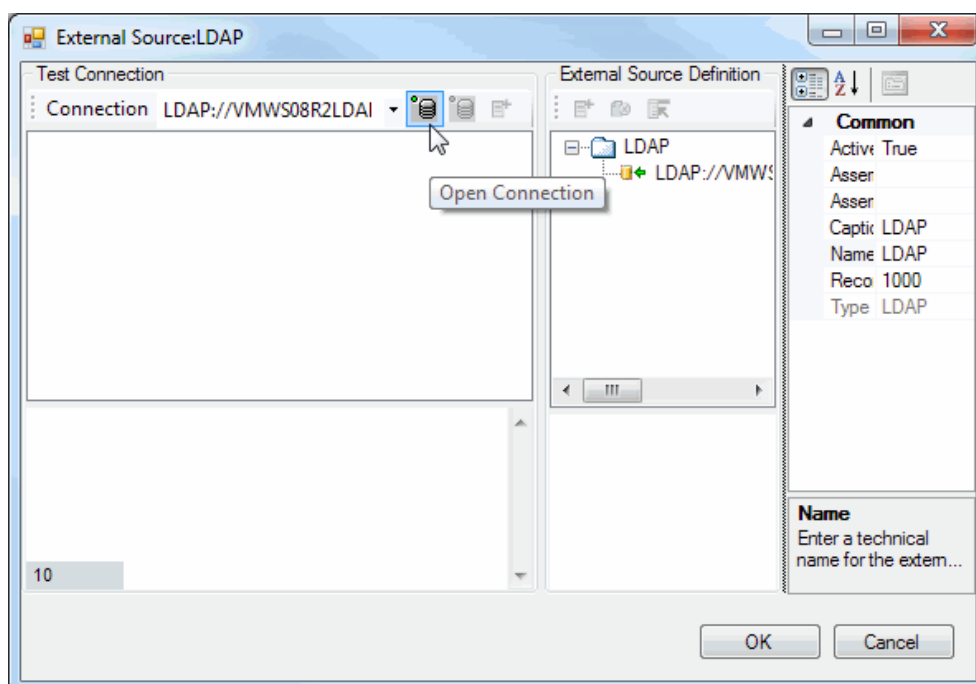
In the **Test Connection** pane, click the **Open Connection** button in the **Test Connection** pane. If your configuration is correct, a connection to the external source is established and you can see the database structure in the **Test Connection** pane. Attributes of external objects are displayed in the window beneath the **Test Connection** pane if you click an object class in the tree.

If multiple connections are available in your external source configuration, you can test them subsequently by selecting a connection in the drop-down field left of the **Open Connection** button.

After a connection to the external database has been established, you can proceed with the next steps directly in the configuration interface working on the nodes in the External Source Definition window. The information displayed in the lower left window about the structure of the external database helps you to correctly map your data. Please note however that only properties of the type string can be handled directly in the configuration interface.

7) Right-click the external source node and select **Add New External Object**.

8) In the selector that opens, select the object class of the Alfabet meta-model that the external class should be mapped to.

9) Click **OK**. A new class node is added to the external source configuration. The name of the node is identical to the name of the object class in the Alfabet meta-model that is mapped to the external data.

10) In the attribute window of the class node, define the attributes required to map the external object class to the selected object class of the Alfabet meta-model:

- **Additional Condition**: Specifies a search query to limit the data displayed to a sub-set of the available external data. You can specify, for example, to import only data for users with a user name starting with "A".

  > Note the following for the definition of an additional condition:
  >
  > - For LDAP, the query must be defined in the LDAP-specific query language. The correct syntax may vary with the type of LDAP server. The following example shows a query defined for a SUN LDAP server:
  >
  >   ```
  >   AdditionalCondition="(&amp;(uid=*)(mail=*))"
  >   ```
  >
  > - For other external source types, the query must be specified as a condition in native SQL, but may not start with a `WHERE` statement. For example:
  >
  >   ```
  >   AdditionalCondition="uid like 'extN%'"
  >   ```
  >
  > - The additional conditions must be written XML-compliant. Thus, you must replace special characters with the respective XML-compliant code (for example, the `&` character must be replaced with `&`).

- **Alfa Deleted Flag Property**: Enter the name of a property of the Alfabet object class that is set to `true` if the object cannot be mapped to a corresponding object in the external data source during a batch synchronization (see below).

  > The **Deletion Requested** property is preconfigured in Alfabet for the class `Person`. For other object classes, a custom property must be configured by the user to store the deletion request. For more information about configuring a custom property, see the section *Configuring Custom Properties for Protected or Public Object Classes* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- **Alfa Properties Creation Values**: If an imported Alfabet object has a property that is not specified in the external data source and is therefore not imported, you can use this parameter to set a fixed value for the property. The syntax is: <name of Alfabet attribute>:<value for Alfabet attribute>. If multiple properties are to be set, the specifications must be comma-separated.

- **External Name**: Enter the name of the external object class that is mapped to the specified Alfabet object class.

- **Create If Not Found**: If set to `True`, an Alfabet object will be created when the external object being accessed (identified in the **Mapping Key** property) does not correspond to an existing Alfabet object. Otherwise, data will only be processed if the data set corresponds to an existing Alfabet object.

11) Right-click the class node in the explorer and select **Add New External Attribute**. A new property node is added to the explorer and the attribute window for the new node opens.

12) Define the following attributes to map a property of the object class from the Alfabet meta-model to a property of the mapped external object class:

- **Alfa Property**: From the drop-down field, select the object class property that should be mapped with the external property.

- **Attribute Type**: Enter the data type of the external attribute. The default data type is `NVARCHAR`.

- **External Caption**: Enter a caption to be displayed in search result tables of selectors in the Alfabet user interface to display data about the mapped property.

- **External Name**: Enter the name of the external property that should be mapped with the property defined with the **Alfa Property** attribute.

- **Is Class Key**: Set to `True` if the property must be unique to identify a set of data. Please note that this attribute can only be set to `True` for one mapped property. It is not possible to define two properties as key.

- **Show Width**: Enter the percentage of available space of selector windows that will be reserved to display this property.

- **Use for Login**: Set to `True` if the property is relevant for user login. For more information about configuring the use of data from an external source for login to the Alfabet user interface, see Configuring the Alfabet Web Application to Perform User Authentication Based On User Data from an External LDAP Data Source.

- **Use for Search**: Set to `True` to allow the Alfabet user to define search conditions for the property in the selector.

- **Use for Show**: Set to `True` to display the property in the search result table of the selector.

- **Use for Sort**: Set to `True` if the property should be used to sort the data sets on the external database server in ascending alpha-numeric order.

  (i) This attribute is only valid if the **Use Sort Options** attribute of the connection configuration is set to `True`.

13) For each additional property of the object class that should be mapped, repeat steps 11 and 12.

14) For each additional class of the Alfabet meta-model that should be mapped to an external object class, repeat steps 7.- 13.

To delete the whole configuration of an external source or parts of a configuration, select the **Delete** option in the context menu of the element that you want to delete.

Configuring the Use of External Source Pools

You can configure an external source pool instead of a single external source to synchronize data from multiple sources.

To configure access and data mapping for a single external data source:

Data sources can alternatively be configured in a source pool. The configuration of a single source in a source pool is identical to the configuration described here. The configuration of the parent source pool is described in the section.

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer and right-click the server alias of the Alfabet Web Application for which you want to configure the interface for data import and select **Connect**.

2) In the explorer, click the **External Sources Configuration** node and select **Add New External Source Pool**. A new external source pool node is added as a suboardinate node to the **External Sources Configuration** node and the attribute window opens in the right pane.

3) Edit the following attributes if applicable:

   • **Name**: Define a unique name for the external source pool. The name is used to identify the external source pool in other parts of the external source configuration such as in the configuration of selectors.

4) Define at least one external source as a sub-element of the external source pool node. To create a new source pool, right-click the external source pool node and select one of the following:

   • **Add New LDAP External Source** to configure synchronization with an LDAP table.

   • **Add New Oracle External Source** to configure synchronization with an Oracle database.

   • **Add New SQL Server External Source** to configure synchronization with a Microsoft SQL Server database.

   • **Add New Access External Source** to configure synchronization with a Microsoft Access database.

   A new external source sub-node is added to the external source node and the attribute window displays the attributes of the external source sub-node.

5) Configure the external source as described in the section Configuring Access to an External Data Source and Mapping of Data.

Configuring the Alfabet Web Application to Perform User Authentication Based On User Data from an External LDAP Data Source

If the object class Person of the Alfabet meta-model is mapped to user data from an external data source, the user authentication data from the external data source can be used to identify the Alfabet user during login to the Alfabet user interface.

When the user first logs in and the user name and password are verified via the LDAP server, a user will be created in Alfabet with a random password for security reasons. The password stored in the Alfabet database is not used for verifying login actions. Only data from the LDAP server is relevant for login. Once a password has been created in the Alfabet database, it will not be changed. The user will be added to the Alfabet database as an Alfabet external user. Actions such as regenerating or resetting a user password, which is performed by a user administrator in the Alfabet user interface will only be executed for Alfabet internal users and will not change the data for LDAP managed users.

The following configuration is required:

- In the external source configuration, the **Use for Login** attribute must be set to `True` for all property mappings of properties used for user authentication. For more information, see Configuring Access to an External Data Source and Mapping of Data.

- Configuration of the remote alias of the Alfabet Clients to use the external data source to authenticate the user. The configuration of the remote alias is described in this section.

  If you want to use the data from the external source for authentication when opening Alfabet Expand with a server alias, the configuration described for the remote alias must be performed in the Client Settings tab of the server alias used for standalone access to Alfabet Expand.

Note the following for the user authentication via an external data source:

- The external data source must be an LDAP server.

- User authentication via an external source will fail if any domain specific authentication is configured in the Alfabet Administrator. For more information about domain specific authentication, see Restricting Authentication to Specific Domains or Specific Users.

- If user authentication is performed via an LDAP external source, and the LDAP server returns multiple `distinguished names` (`DN` names) for the same login, the login data of the user is tested against all combinations of `DN` name and password and is allowed to log in if any of the combinations returns a match. The user data for the user with the matching `DN` name and password combination is returned from the external LDAP source and used to update or create the user data for the user in the Alfabet database, if applicable.

- If user authentication is performed via an LDAP external source, and multiple `ID`s are available for one `distinguished name` (`DN` name), the Alfabet Web Application can be configured to store the set of ID values in a custom attribute defined by the customer for the class Person of the Alfabet meta-model. During login, the check for correct user name is performed against all `ID`s in the string array and if one of the combinations of `ID` and password matches, the user is allowed to log in to the Alfabet user interface.

  > Multiple `ID`s are typically configured in LDAP to identify users from different contexts. For example, while internal personnel is identified by the Windows `ID`, external users (for example, customers or partners) are identified via their email address.

- Asterisks and accents are not allowed in the login name.

- A restriction of the maximum number of consecutive failed login attempts is not possible if authentication is managed via an external data source such as for an LDAP server. For more information about authentication via an external data source, see Tracking Login Actions in the Windows Event Log.

To configure the remote alias to use authentication via an external database:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer in the Alfabet Administrator, The right pane displays a list of all available alias configurations.

2) In the table, select the remote alias configuration that you want to edit.

   > The following settings can also be performed in the **Client Settings** tab of the server alias configuration. The configuration of the server alias is required if you want to use

the external source for authentication when logging in to Alfabet Expand in standalone mode.

3) In the toolbar, click the **Edit** 🖊 button. You will see the editor in which you can edit the alias configuration.

4) Go to **Client Settings** > **Authentication** tab to open it and define the following settings:

- **Mode**: Select `ExternalSource` in the drop-down field to enable the authentication via data from an external LDAP table.

- **External Source**: Enter the name of the external source or external source pool that shall be used for user authentication. The name is specified with the **Name** attribute of the external source (pool) in the external source configuration. (For more information about the configuration of the external source, see Configuring Access to an External Data Source and Mapping of Data.

5) Click **OK** to save your changes.

Configuring Authentication Based On Login with One of Multiple Permissable LDAP IDs

To configure identification of a user based on one of a set of LDAP `ID`s, the following configuration is required in addition to the basic configuration described above:

- In the configuration tool Alfabet Expand, add a new custom property to the object class `Person`. The **Property Type** attribute of the custom property must be set to `String`. For more information about the creation of custom properties for an object class, see *Configuring Custom Properties for Protected or Public Object Classes* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- In the remote alias configuration of the Alfabet Clients, activate and configure the storage of the compound user IDs as described below.

To configure the remote alias to fill in the user defined compound user ID property of the object class `Person`:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer. The right pane displays a list of all available alias configurations.

2) In the table, select the remote alias configuration that you want to edit.

3) In the toolbar, click the **Edit** 🖊 button. You will see the editor in which you can edit the alias configuration.

4) Go to **Client Settings** > **Actualization** tab to open it and define the following settings:

- **Mode**: Select `ExternalSource` in the drop-down field to enable the actualization of the compound user ID from the external LDAP table. Update is performed during login of a user for the compound user ID of the user logging in.

- **Compound User ID**: Enter the name of the custom property for the object class `Person` that should be used to store the compound user ID.

5) Normally the update of the compound user ID is performed using the data of the same external source that is used for authentication of user data. In the exceptional case that different external

LDAP servers are used for user authentication and actualization of the compound user ID, the following settings are also required in the **Actualization** tab:

- **External Source**: Enter the name of the external source or external source pool that shall be used for user authentication. The name is specified with the **Name** attribute of the external source (pool) in the external source configuration. (For more information about the configuration of the external source, see Configuring Access to an External Data Source and Mapping of Data).

6) Click **OK** to save your changes.

Configuring the Alfabet Web Application to Perform User Authorization Based On User, User Group and User Profile Data from an External Data Source

Whether a Alfabet user is authorized to view or edit an object in the Alfabet database depends on the access permission concepts implemented in the Alfabet database. Some of the permission concepts depend on relations between the object class `Person` and access permission related to user group (class `UserGroup`) or user profile (class `ALFA_USERPROFILE`).

> An overview of the access permission concepts for access to objects is provided in the section Configuring User Authentication.

The assignment of users to user groups and user profiles can be maintained in an external data source such as for an LDAP database. The relations defined in the external source can be used to update the `RELATIONS` table in the Alfabet database. During update, the external relations are either added to the existing relations in the `RELATIONS` table or substitute the defined user mapping to user profiles and user groups in the Alfabet database with the user mapping defined in the external source.

> User authorization via an external source requires that the configuration of relations between user, user profile and user group in the external LDAP source meets the following conditions:
> - The user is not member of any sub-group.
> - The user is directly linked to the user profile.

User authorization via an external source can be performed when a user logs in to the Alfabet user interface via one of the following authentication mechanisms:

- Standard Login
- External source based login
- Windows sign-on based login
- Federated Authentication

The user data can be read from an LDAP database as well as from a SAML identity provider.

Relations import from the external source is a two-step mechanism:

- Step One: The external relations for the user are written to the database. This step can be performed in two ways:

- The data from the external database are written to the database table
  `ALFA_EXTERNAL_RELATIONS` during login of the user using the Actualization functionality of
  the interface for data synchronization with external sources.

- The data from the external database is written to a database table created during data import
  using the Alfabet Data Integration Framework (ADIF). When ADIF is used for data update, data
  update is independent of user authentication. The table `ALFA_EXTERNAL_RELATIONS` should
  not be used for storage of external data in this scenario because other mechanisms such as
  updating a compound user ID may require data update in the table overwriting the data
  imported via ADIF.

- Step Two: The `RELATIONS` table of the Alfabet database is updated with data from the
  `ALFA_EXTERNAL_RELATIONS` table. Depending on the configuration, relations are either added to
  the existing configuration or all relations between the current user and the relevant object classes
  handled via the external data source are deleted from the relations table and new relations are
  defined on basis of the data in the `ALFA_EXTERNAL_RELATIONS` table.

The following configuration is required:

For data import to the database table `ALFA_EXTERNAL_RELATIONS` the following configuration
is required:

- Configure user authentication via the external source. For more information about the
  required configuration steps, see Configuring the Alfabet Web Application to Perform
  User Authentication Based On User Data from an External LDAP Data Source.

- Configuration of the XML object ***ExternalSourceConfiguration*** to map data from the
  external source to the `ALFA_EXTERNAL_RELATIONS` table. For more information, see
  Configuring the XML Object ExternalSourceConfiguration to Support User
  Authorization.

- Configuration of the relevant Alfabet object classes to allow mapping with the data
  from the external database. For more information, see Configuring the XML Object
  ExternalSourceConfiguration to Support User Authorization.

For more information about data import via ADIF, consult the reference manual *Alfabet Data In-
tegration Framework*

For update of the `RELATIONS` table of the Alfabet database with the data from the table contain-
ing the external data:

- Configuration of the update of the `RELATIONS` table in the XML object
  AuthConfiguration or a copy thereof.

- Configuration of the remote alias of the Alfabet clients to use the external data source
  for user authorization. For more information, see Activating User Authorization in the
  Remote Alias of the Alfabet Client.

Configuring the XML Object ExternalSourceConfiguration to Support User Authorization

The mapping of data required to fill the `ALFA_EXTERNAL_RELATION` table of the Alfabet database with data
from the external data source is defined in the XML object ***ExternalSourceConfiguration***.

The following columns of the `ALFA_EXTERNAL_RELATION` table are relevant for data mapping and must be defined using the XML object configuration:

- `FROMEXTID`: The attribute of the external object class storing user data that is defined as relevant for user login mechanisms in the external source configuration.

- `TOEXTID`: An attribute of the related external class. If not otherwise specified in the external source configuration, the key attribute defined in the external source configuration to identify objects of the related external class is written to this column of the relation table.

- `RELATIONNAME`: A name to identify the relation. The name is a distinguished name defined in the external source configuration for each relation.

The XML object **ExternalSourceConfiguration** must include the required information to fill in the `ALFA_EXTERNAL_RELATION` table. The table is then filled in with the relevant data when a user logs in to the Alfabet user interface and user authentication is performed via the external data source.

The configuration cannot be added to the XML object via the explorer nodes of the **External Sources Configuration** sub-node of the server alias but must be defined by editing the XML object in a text editor.

> (i) The following workflow assumes that the connection to the external source and the mapping of object classes and object class properties is already available in the XML object. For more information about the general definition of an external source connection, see Configuring Access to an External Data Source and Mapping of Data.

To configure user authorization in the XML object **ExternalSourceConfiguration**:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer.

2) Right-click the server alias of the Alfabet Web Application for which you want to configure the XML object and select **Connect**.

3) In the explorer, click the **XML Objects** node.

4) In the table on the right side of the explorer, select **ExternalSourceConfiguration** and click the **Edit** ✎ button in the toolbar. An editor opens displaying the current configuration of the external source.

5) In the XML element `ExternalObject` map the user data in the relevant external source to the user data in the Alfabet database, enter a child XML element `ExternalRelation` for each relation that you want to define for the object class `Person`. For example, if you want to include relations to user groups and user profiles, you must define twoXML element `ExternalRelation`. In the XML element `ExternalRelation` define the following XML attributes:

   - `ExternalClass`: Enter the name of the external class that the external class defining user data is related to.

   - `ExternalAttribute`: Define the attribute of the target external class for the relation used to fill the column `TOEXTID`. If the external data source is an LDAP server, this attribute cannot be specified and the base `DN` of the object is returned.

     > The external attribute values must be identical to the values of a property of the target object class in the Alfabet database. One way to perform this mapping is to define a custom property for the target object class in the Alfabet database and fill it with the values of the external attribute used for external relation definition.

For the Alfabet object classes `Person`, `UserGroup` and `ALFA_USERPROFILE`, a standard attribute `EXTERNAL_ID` is available that can be used to store the mapping information.

You can use the ADIF interface to fill the custom property with the relevant values or define the mapping via a custom editor in the Alfabet user interface.

- `ExternalQuery`: Define a query using the query language applicable for the external source that defines the relation between the external class storing user data and the target external class.

- `RelationName`: Define a unique name for the relation to identify the relation type in the `ALFA_EXTERNAL_RELATION` table. The name can be any string.

The following example shows the configuration of the XML element `ExternalRelation` in the XML element `ExternalSource` defining data synchronization with an LDAP external source. The XML element `ExternalRelation` defines the assignment of users to user groups. The external class is a custom class defined for the LDAP source.

```xml
<ExternalSourcesConfiguration>

    <ExternalSource Name="LDAP" Active="true" Type="LDAP"
    Caption="LDAP">

        <Connection
        ConnectionString"LDAP://LDAPServer/dc=lan"
        AuthenticationMode="ServerBind"/>

        <ExternalObject AlfaClass="Person"
        ExternalName="person">

            <ExternalAttribute ExternalName="uid"
            ExternalCaption="USER NAME"
            AlfaProperty="USER_NAME" UserForLogin="true"
            IsKey="true" UserForShow="true"/>

            <ExternalRelation ExternalClass="alfagroup"
            ExternalQuery="(member=:USERDN)"
            RelationName="UserToGroup"/>

        </ExternalObject>

    </ExternalSource>

</ExternalSourcesConfiguration>
```

- Click `OK` to save your changes.

Configuring the Update of Alfabet Internal Relations in the XML Object AuthConfiguration

The XML object *AuthConfiguration* specifies the mapping between the `ALFA_EXTERNAL_RELATION` table and the `RELATIONS` table of the Alfabet database. The XML object can be edited in the tool Alfabet Expand or in the tool Alfabet Administrator. If you want to define different mappings for different Alfabet Clients, you can create copies of the XML object *AuthConfiguration* in the tool Alfabet Expand and define different mappings in each of the copies.

To edit the XML object *AuthConfiguration* in the tool Alfabet Expand:

1) Expand the **XML Objects** node in the explorer of the **Presentation** tab of Alfabet Expand.

2) Expand the **Administration** node.

3) Right-click *AuthConfiguration* and select **Edit XML** in the context menu. The XML object opens in the center pane.

4) Define the required XML elements in the editor.

5) If the XML object shall be visible and editable in the Alfabet Administrator, set the **Visible in Administrator** attribute in the attribute window of the XML object *AuthConfiguration* to `True`.

6) In the menu of Alfabet Expand , click the Save 💾 button to save your changes.

To create a copy of the XML object *AuthConfiguration* in the tool Alfabet Expand and define the configuration in the new XML object:

1) Expand the **XML Objects** node in the explorer of the **Presentation** tab of Alfabet Expand.

2) Expand the **Administration** node.

3) Right-click *AuthConfiguration* and select **New XML Object As Copy** in the context menu. A new XML object *AuthConfiguration_1* is added to the explorer.

4) Click the new XML object node in the explorer and optionally edit the following attributes in the attribute window:

   - **Name**: If applicable, edit the technical name of the XML object. The name must be unique and may not contain whitespaces or special characters.

   - **Visible In Administrator**: Set the attribute to `True` if you want the XML object to be visible and editable in the tool Alfabet Administrator.

5) Right-click the new XML object in the explorer and select **Edit XML** in the context menu. The XML object opens in the center pane.

6) Define the required XML elements in the editor.

7) In the menu of Alfabet Expand , click the Save 💾 button to save your changes.

To edit the XML object in the tool Alfabet Administrator:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer.

2) Right-click the server alias of the Alfabet Web Application for which you want to configure the XML object and select *Connect*.

3) In the explorer, click the **XML Objects** node.

4) In the table on the right side of the explorer, select *AuthConfiguration* or, if your configuration is based on a copy of the XML object *AuthConfiguration*, the relevant copy, and click the

   Edit 📝 button in the toolbar. An editor opens displaying the current configuration of the XML object.

   > ⓘ If you do not see the XML object that you want to edit in the table, the **Visible in Administrator** attribute of the XML object is set to `False`. The attribute must be set to `True` using the tool Alfabet Expand to make the XML object visible in the Alfabet Administrator. The setting of attributes for XML objects in Alfabet Expand is explained above.

5) Define the required XML elements in the text editor.

6) Click **OK** to save your changes.

The XML object must consist of a root XML element ***AuthConfiguration*** containing a child XML element ***Authorization*** that is a container for one or multiple XML elements ***AuthQuery***. For each relation that shall be updated on basis of the data in the `ALFA_EXTERNAL_RELATION` table, an XML element ***AuthQuery*** must be added to the XML object.

The following attributes can be defined in the elements of the XML element ***AuthQuery***:

XML Attributes of the XML Element AuthQuery	Required Configuration
`Mode`	Set this XML attribute to one of the following values: • `Add`: Existing relations in the Alfabet`RELATIONS` table that does not exist in the `ALFA_EXTERNAL_RELATION` table are not deleted. If a relation exists in both tables, it is kept. If it exists in the `ALFA_EXTERNAL_RELATION` table only, it is added to the `RELATIONS` table. • `Replace`: All existing relations of the defined type are deleted from the Alfabet`RELATIONS` table prior to adding relations from the `ALFA_EXTERNAL_RELATION` table.
`TargetProp-erty`	Define the name of the property of the object class `Person` that defines the relation to the target object class. Only relations defined via the specified property for the user currently logged in are included in the data update process.
`Query`	Define an Alfabet query or native SQL query that finds the target objects of the relation in the Alfabet database. In the `RELATIONS` table, relations are defined via three columns: • `FROMREF`: Defines the `REFSTR` of the object class referencing another object class. If a new relation is added to the `RELATIONS` table on basis of an XML object ***AuthConfiguration*** definition, the `FROMREF` column is filled with the `REFSTR` of the user currently logging in. • `PROPERTY`: Defines the name of the property of the `FROMREF` object class that stores the relation to the target object class. If a new relation is added to the `RELATIONS` table on basis of an XML object ***AuthConfiguration***, the `PROPERTY` column is filled with the value of the XML attribute `TargetProperty` of the relevant XML element `AuthQuery`. • `TOREF`: Defines the `REFSTR` of the target object for the relation. The query defined with the XML attribute `Query` of the XML element `AuthQuery` must return the `REFSTR` of the Alfabet object that is target of the relation. For each object found, a relation is added to the `RELATIONS` table. To find the relevant target objects in the Alfabet database, mapping of the data in the

XML Attributes of the XML Element Au- thQuery	Required Configuration
	`ALFA_EXTERNAL_RELATION` table to objects in the Alfabet database must be defined via the query.

The following example shows an XML object ***AuthConfiguration*** defining update of the assign- ment of a user to user groups. The example is based on the following:

- The property `UserGroups` of the Alfabet object class `Person` stores the relation to objects of the object class `UserGroup`. This property is defined via the XML attribute `TargetProperty`.

- The standard property `EXTERNAL_ID` of the Alfabet object class `UserGroup` is used to define the mapping to the values provided in the `TOEXTID` column of the `ALFA_EXTERNAL_RELATION` table.

- The standard property `EXTERNAL_ID` is defined for the Alfabet object class `Person` to map this class with the data provided in the `FROM_EXTID` column of the `ALFA_EXTERNAL_RELATION` table.

- The parameter `CURRENT_USER` of the Alfabet query language is used in the queries to identify the user that currently logs in.

- The `WHERE` clause in the Alfabet query limits the definition of relations to relations defined for the current user and relations reflecting mapping to user groups in the `ALFA_EXTERNAL_RELATION` table. These relations are identified by the `RELATIONNAMEUserToGroup`.

```
<AuthConfiguration>

    <Authorization>

        <AuthQuery Mode="Add" TargetProperty="UserGroups"

        Query="ALFABET_QUERY_500 FIND UserGroup

            INNERJOIN ALFA_EXTERNAL_RELATION ON UserGroup.EXTERNAL_ID
            = ALFA_EXTERNAL_RELATION.TOEXTID

            INNERJOIN Person ON Person.EXTERNAL_ID =
            ALFA_EXTERNAL_RELATION.FROMEXTID

            WHERE (AND

                Person.RefStr CONTAINS:CURRENT_USER

                ALFA_EXTERNAL_RELATION.RELATIONNAME='UserToGroup')"

        />

    </Authorization>

</AuthConfiguration>
```

In Alfabet, user profile assignment depends on the status of the user. If the status is Anonymous, the user interface automatically opens with the viewer profiles marked as standard profile for anonymous user. If the status is `NamedUser`, user profiles must be assigned to the user. The user can then log in with any of the assigned user profiles. If no user profiles are assigned to a named user, the user cannotaccess the Alfabet user interface.

If user profiles are assigned to an anonymous user via the query definition in the XML object ***AuthConfiguration***, the status of the user is automatically changed to `NamedUser`. But if the user profile definition for the user is removed in the external LDAP source and the update of the user profiles via `AuthConfiguration` leads to a detachment of all user profiles from the `NamedUser` in the Alfabet database, the user is not changed back to Anonymous user. If you want the user to access Alfabet with the viewer profile for anonymous access, you must define the query in the XML object `AuthConfiguration` to assign the view profile to all users or to all users for which no user profiles have been assigned in the external database.

Activating User Authorization in the Remote Alias of the Alfabet Client

To configure the remote alias to use authorization via relations stored in an external database:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer. The right pane displays a list of all available alias configurations.

2) In the table, select the remote alias configuration that you want to edit.

3) In the toolbar, click the **Edit** 🖉 button. You will see the editor in which you can edit the remote alias configuration.

4) Go to the **Client Settings** > **Authorization** tab to open it.

5) Enter the name of the XML object containing the authorization configuration in the **XML Object** field. The name is either `AuthConfiguration` or the name defined with the **Name** attribute of the copy of the XML object ***AuthConfiguration***.

6) Go to the **Client Settings** > **Actualization** tab and define the following settings:

   • **Mode**: Select `ExternalSource` in the drop-down field to enable the actualization of the compound user ID from the external LDAP table. Update is performed during login of a user for the compound user ID of the user logging in.

7) Normally actualization is performed using the data of the same external source that is used for authentication of user data. In the exceptional case that different external LDAP servers are used for user authentication and actualization, the following settings are also required in the **Actualization** tab:

   • **External Source**: Enter the name of the external source or external source pool that shall be used for actualization. The name is specified with the **Name** attribute of the external source (pool) in the external source configuration. (For more information about the configuration of the external source, see [Configuring Access to an External Data Source and Mapping of Data](#).

8) Click **OK** to save your changes.

Chapter 9:   Batch Deletion of Objects from the Alfabet Database

The executable `AlfaDeleteConsole.exe` allows you to delete data in console mode.

Such delete jobs are complex in Alfabet. Several links must be rescanned and dependent objects must be deleted together with the main object.

Delete jobs are available for the following Alfabet object classes:

- `Application`
- `ICTObject`
- `OrgaUnit` (Organization)
- `BusinessProcess`
- `Assignment`
- `Person`

The import of data will be triggered by a batch utility:

Executable	`AlfaDeleteConsole.exe` located in the ***Programs*** subdirectory of the Alfabet installation directory
Remote or stand-alone access	Stand-alone access with a server alias
Preconditions	See the section Preconditions for Performing Delete Jobs.
Logging	Standard Alfabet logging. For information about standard logging and the command line options, see Standard Logging for Alfabet Batch Utilities.
Command line help	Start executable with `-h` or `-help`

Preconditions for Performing Delete Jobs

To perform delete jobs, at least one configuration file must be located in the same directory as the executable `AlfaDeleteConsole.exe`:

- **Mandatory:** `AlfabetMS.xml` file with a server alias configuration to connect to the Alfabet database. For information on how to configure the `AlfabetMS.xml`, see Creating a New Server Alias. The `AlfabetMS.xml` for the Alfabet Server can be used for the executable without changes. If `AlfaDeleteConsole.exe` is located in another directory, the path to the AlfabetMS.xml file can be specified in the command line.

- **Optional:** `AlfaDeleteConsoleConf.xml` file for the definition of statements controlling the execution process. The statements in this file define which data will be deleted. If this configuration file is not available, the command line for execution of `AlfaDeleteConsole.exe`

has to contain the specification of the data which should be deleted The file can be renamed. The new name must then be specified in the command line when execution `AlfaDeleteConsole.exe`.

⚠️ The Alfabet Server must be shut down to execute the `AlfaDeleteConsole.exe`.

Using AlfaDeleteConsole.exe with the alfaDeleteConsoleConf.xml Configuration File

Prior to deleting data with `AlfaDeleteConsole.exe`, the specification of the data which must be deleted must be written to the `alfaDeleteConsoleConf.xml` configuration file. You can use any standard text editor to edit the file.

The file can contain one or multiple `AlfaDeleteStatements` in the following syntax:

```
<AlfaDeleteStatement

    MSAlias="Alfabet"

    StatementName="Delete_Applications_for_DEP1"

    ClassNames="ICTObject,Application"

    Query="ALFABET_QUERY_500 FIND Application WHERE (AND Name like 'DEP1_%'
    LAST_UPDATE&lt;'2004/10/16')"

/>
```

The following applies to the XML attributes for the XML element `AlfaDeleteStatement`:

`alias`	The server alias of the configuration in the `AlfabetMS.xml` configuration file which should be used by the `AlfaDeleteConsole.exe` to connect to the database.
`State-mentName`	Name of the `AlfaDeleteStatement`. This name is used in the command line to start the `AlfaDeleteConsole.exe` to select the definitions for deletion.
`Class-Names`	The Alfabet object class for which data is deleted. Use a comma-separated format to specify more than one class.
`Query`	The definition of data which will be deleted in a complete Alfabet query using Alfabet query language 500.
	NOTE: The XML attribute `Query` attribute is only valid if the defined Alfabet query contains no show and sort properties.
	For information on Alfabet query syntax, see the reference manual *Configuring Alfabet with Alfabet Expand*.

To execute the `AlfaDeleteConsole.exe`, use the following command line statement:

```
AlfaDeleteConsole.exe-statementName <statement name from
AlfaDeleteConsoleconf.xml>-alfaLoginName <user name>-alfaLoginPassword <user
login password> [-defFile <Definition File>]
```

The table below displays the command line options:

Command Line Option	Manda-tory/Op-tional	Explanation
`-msalias` <server alias>	Mandatory	Name of the server alias to access the Alfabet database.
`-msaliasesfile` < Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` file that contains the specification of the server alias is not located in the same directory as the executable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName`	Mandatory	User name for login to the database. ⚠ A user can only execute a batch job if the **Can Execute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPassword`	Optional	Password for login to the database.
`-statementName` <name of statement in AlfaDeleteConsole-Conf.xml>	Mandatory	Name of the `AlfaDeleteStatement` in the `AlfaDeleteConsoleConf.xml` file that should be executed with this delete job.
`-defFile` <definition file>	Optional	If the definition file is not named `AlfaDeleteConsole-Conf.xml`, the file name must be specified with this parameter.

Using AlfaDeleteConsole.exe with Data Specification in the Command Line

To execute the `AlfaDeleteconsole.exe`, use the following command line statement:

```
AlfaDeleteconsole.exe-msalias <server alias>-classes <ClassNames>-query
"<Alfabet query>"
```

The table below displays the command line options:

Command Line Option	Manda-tory/Op-tional	Explanation
`-msalias` <server alias>	Mandatory	Name of the server alias to access the Alfabet database.
`-msaliasesfile` < Alfabet configuration file path>	Optional	If the `AlfabetMS.xml` file that contains the specification of the server alias is not located in the same directory as the exe-cutable, the path to the `AlfabetMS.xml` file must be specified with this parameter.
`-alfaLoginName`	Mandatory	User name for login to the database. ⚠ A user can only execute a batch job if the **Can Exe-cute Batch Jobs** checkbox is selected (=`True`) for the user.
`-alfaLoginPassword`	Optional	Password for login to the database.
`-classes` <class names>	Mandatory	The Alfabet object class for which data is deleted. Use a comma-separated format to specify more than one class.
`-query`	Mandatory	The definition of data which will be deleted in a complete Alfabet query using Alfabet query language 500. **NOTE**: The Query attribute is only valid if the defined Alfabet query contains no show and sort properties. For information on Alfabet query syntax, see the *Configuring Alfabet with Alfabet Expand* Reference Manual.

Chapter 10: Working with the Alfabet Administrator

The Alfabet Administrator allows you to configure the Alfabet components and to perform database-related tasks such as restoring databases and updating the meta-model.

This chapter describes all functionalities available in the Alfabet Administrator. The following description is not workflow-oriented. Workflow-oriented descriptions of Alfabet Administrator functionalities are included in the previous chapters of this manual.

Accessing the Alfabet Administrator

The Alfabet Administrator will be installed automatically together with the Alfabet components. The `AlfaAdministrator.exe` is located in the Alfabet`Programs` directory and can be accessed in the Windows® **Start** menu of the Alfabet components host.

Understanding the Interface of the Alfabet Administrator

The Alfabet Administrator shows an explorer in the left pane. The explorer has two nodes below the root node:

- **Alfabet Aliases**

  If you expand the **Alfabet Aliases** explorer node, you will see all alias configurations that are available in the `AlfabetMS.xml` configuration file in the `Programs` directory of your Alfabet installation. You can create new aliases, edit existing alias configurations, and perform several administrative actions on the Alfabet database.

- **Usage Tracking**

  If you click the explorer node, you can view a history of the use of Alfabet functionalities by users with a specified user profile. Usage tracking is only available if the Alfabet Server is configured to track the use, which is required to fulfill the specifications of metered contracts.

When you right-click a node in the explorer, a context menu opens. The content of the menu depends on the node you are currently working with. Some functionalities in the context menu allow access to additional parts of the explorer that are only displayed after the right-click action is performed. For example, when selecting **Connect** in the context menu of an alias, the alias node in the explorer is expanded and sub-nodes that represent configuration steps affecting the Alfabet database are displayed in the explorer.

When you click a node in the explorer, the content of the work area at the right side of the explorer changes to allow working with the respective functionality.

Changing the Language of the Alfabet Administrator

By default, the interface of the Alfabet Administrator is rendered in English. If you want to work with the Alfabet Administrator in any other language culture implemented by your company, you can change the language by selecting a language culture from the language selector in the upper-right corner of the user interface of the Alfabet Administrator.
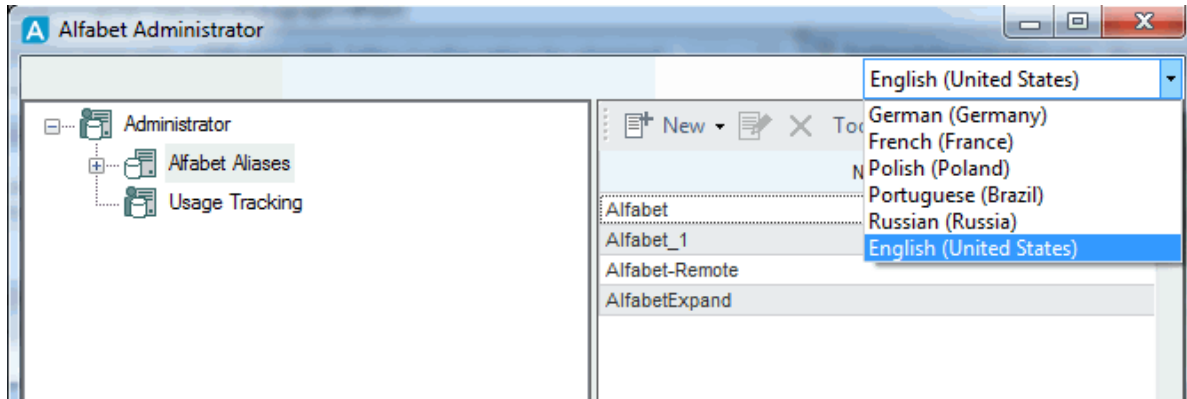
Working with Alfabet Aliases

The main configuration of all Alfabet components is done via alias configurations. There are two different types of alias configurations:

- Server alias configurations for all components that directly connect to the Alfabet database. A server alias configuration includes the parameters required for the connection to the Alfabet database, define, which authentication mode is used for user authentication and specify parameters for all relevant components such as for the default language setting for the user interface.

- Remote alias configurations for components with indirect access to the Alfabet database via the Alfabet Server. The remote alias configuration specifies the parameters required for the connection to the Alfabet Server and component specific parameters.

The following information is available about working with the alias configurations in the Alfabet Administrator:

- Creating and Editing Aliases

    - Creating a New Server Alias

    - Creating a Server Alias as Copy of an Existing Server Alias

    - Creating a Remote Alias

    - Editing an Existing Server Alias or Remote Alias

- Configuration Attributes for the Alfabet Components

    - Defining Connections Based On Server Variables

- Deleting an Alias Configuration

- Copying Existing Alias Configurations to a Separate AlfabetMS.xml File in Another Directory

- Exporting a Report about an Alias Configuration

- Sorting the Alias Nodes in the Explorer Tree

Creating and Editing Aliases

Alias configurations are created and edited with the Alfabet Administrator. The Alfabet Administrator writes the alias definitions to the configuration file `AlfabetMS.xml` that is located in the same directory as the Alfabet Administrator executable. A configuration file can include multiple alias configurations. When you start an Alfabet component, you must define the alias that shall be used for connection to the Alfabet database or to the Alfabet Server.

Creating a New Server Alias

Do the following to add a server alias configuration to the `AlfabetMS.xml` file:

1) Click the **Alfabet Aliases** node in the **Administrator** explorer. The right pane displays a list of all available alias configurations.

2) In the toolbar, select **New** > **Create Alias**.

3) In the editor that opens, configure the settings for the server alias configuration. For a new server alias configuration, many fields in the editor will be prefilled with best practice settings. For an explanation of the fields in the editor, see the section Configuration Attributes for the Alfabet Components.

4) Click **OK** to save your changes.

Creating a Server Alias as Copy of an Existing Server Alias

When you define multiple server alias configurations for Alfabet components that connect to the same Alfabet database but are using different user authentication mechanisms, for example, you can define one basic server alias and define all other server alias configurations as copy of the first server alias. The configuration of the basic server alias is written to the copy and can be altered according to your demands.

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer and right-click the server alias that you want to create a copy from and select **Create the Alias as Copy**.

2) In the editor that opens, configure the server alias settings. For an explanation of the fields in the editor, see the section Configuration Attributes for the Alfabet Components.

3) Click **OK** to save your changes.

Creating a Remote Alias

A remote alias is a client configuration for an existing server alias. Do the following to add a remote alias configuration for an existing server alias.

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer and right-click the server alias that you want to create a remote alias for and select **Create Remote Alias**.

2) In the editor that opens, configure the client settings. For an explanation of the fields in the editor, see the section Configuration Attributes for the Alfabet Components.

3) Click **OK** to save your changes.

Editing an Existing Server Alias or Remote Alias

Do the following to edit an alias configuration:

ⓘ   Make sure that the server alias is not connected. Settings of a connected server alias cannot be changed.

1) Click the **Alfabet Aliases** node in the **Administrator** explorer. The right pane displays a list of all available alias configurations.

2) In the table, select the alias configuration that you want to edit.

3) In the toolbar, click the **Edit** 🖉 button. You will see the editor in which you can edit the alias configuration.

4) Enter the information in each field, as required. For an explanation of each field, see the section Configuration Attributes for the Alfabet Components.

5) Click **OK** to save your changes.

Configuration Attributes for the Alfabet Components

The following attributes can be edited in the Alfabet Administrator for the configuration of the server alias and the remote alias. In the remote alias editor, the attributes that are not relevant for a remote alias are ReadOnly:

Parameter	Description
Overview tab	
Name	Enter the alias name for the configuration set. The alias name is used to identify the configuration in the login dialog box displayed when starting one of the Alfabet components, for example. ⚠ The name of the alias must be unique within the `AlfabetMS.xml` file. If your `AlfabetMS.xml` contains two aliases with the same name, both configurations are invalid. ⚠ The name of the server alias should be short. The alias name is used to create a path to a temporary directory for storing Alfabet component-related files during operation of the Alfabet components. The length of the file names for the temporary files including the path information must not exceed 255 characters.
Is Remote	This value specifies whether you want to configure a server alias or a remote alias. To configure a remote alias, select the checkbox.

Parameter	Description
	Alfabet components started with a server alias configuration are directly connecting to the Alfabet database. Alfabet components started with a remote alias configuration connect to the Alfabet database via an Alfabet Server. **NOTE**: You cannot define a server alias based on an existing remote alias by deselecting the **Is Remote** checkbox.
Host	This attribute is only relevant when an Alfabet Server is used to run tools that require an indirect connection to the Alfabet database via the Alfabet Server. Enter the fully qualified domain name or IP address that the Alfabet Server is installed on. For the server alias or a remote alias that is started on the same host as the Alfabet Server, the loopback address (127.0.0.1) should be specified instead of the IP address of the server host.
Port	This attribute is only relevant when an Alfabet Server is used to run tools that require an indirect connection to the Alfabet database via the Alfabet Server. Enter either the port number of the communication port used by the Alfabet Server to listen for incoming TCP connections (server alias), or the port that the client application connects to (remote alias).
Server	This attribute is only relevant when an Alfabet Server is used to run tools that require an indirect connection to the Alfabet database via the Alfabet Server. Name of the server object. The server alias and corresponding remote aliases must use the same setting.
Ensure Security	If the machines on which the Alfabet components are installed are members of the same active directory, you can select the checkbox to use.NET EnsureSecurity to secure the communication channel to the Alfabet Server. This setting must be the same on all participating machines (server alias and corresponding remote aliases).
Help Server	The URL to access the Alfabet online Help files. For a typical installation, enter: http://webserver/ Alfabet _virtual_directory/help. For local installations, the absolute path to the directory containing the online Help files should be specified in the following syntax: `file:///`<absolute path to the folder containing the Alfabet online help folder Gui_30 as sub-folder> Server variables can be used to define part of the information as a variable with the value set in the **Variables** tab of the server alias configuration. For more information, see Defining Connections Based On Server Variables.
Web Server	The base URL used for creating express views and links in emails sent by the Alfabet Web Application or the Alfabet Server. This setting is required to ensure full functionality of the Alfabet user interface. Enter the full Web server name and the name of the

Parameter	Description
	application directory created during the basic installation. A typical entry would be: http://webserver/Alfabet_virtual_directory Server variables can be used to define part of the information as a variable with the value set in the **Variables** tab of the server alias configuration. For more information, see Defining Connections Based On Server Variables.
Search Index Directory	The URL to access the search indexes needed for the full-text search functionalities for the online Help and Alfabet solution. The property should be set to the absolute path of the directory that contains the search indexes (defined during the set up process). The property should be different for each installation on the same machine. Server variables can be used to define part of the information as a variable with the value set in the **Variables** tab of the server alias configuration. For more information, see Defining Connections Based On Server Variables.

Expand tab

Parameter	Description
Track Meta-Model Changes	Select the changes if you want the configuration tool Alfabet Expand to scan the Alfabet meta-model for changes in the interval defined with the **Tracking Period** attribute. This feature should be activated when multiple instances of Alfabet Expand are used in parallel with a connection to the same Alfabet database. If the meta-model is changed in one of the instances of Alfabet Expand and the changes are saved to the meta-model, the users of the other Alfabet Expand instances will be informed via a message on the Alfabet Expand host that the meta-model has changed. Clicking the message opens a report in the center pane of Alfabet Expand that provides information about which change was made to the meta-model and the user that performed the change. The report is displayed in all instances of Alfabet Expand started with an alias for which the **Track Meta-Model Changes** attribute is activated.
Tracking Period (seconds)	Select the time period to track changes to the meta-model in seconds.
Test Web Application	Enter the URL of a running Alfabet Web Application that shall be used to open the Alfabet user interface from within the Alfabet Expand application. The path must start with http:// or https:// and end with /. Server variables can be used to define part of the information as a variable with the value set in the **Variables** tab of the server alias configuration. For more information, see Defining Connections Based On Server Variables.
Alias for Meta-Model Design Master Database	This option is only recommended for test environments. If two different Alfabet database s are used for configuring the solution and testing the application, the changes made to the database of the solution configuration environment can be uploaded to the database of the test environment via a direct connection between Alfabet Expand of the test environment with the Alfabet database of the solution configuration environment.

Parameter	Description
	For this scenario, the server alias used for Alfabet Expand in the test environment must be configured to connect to the database of the solution configuration environment.
	A server alias that connects to the Alfabet database of the solution configuration environment must be selected in the drop-down field of the **Alias for Meta-Model Design Master Database** attribute.
	This option is only taken into account if the functionality is activated via the **Enable Master Database Configuration** attribute. For a description of the complete configuration, see Configuring Alfabet Expand to Update the Configuration from a Master Database via a Direct Connection.
Update from Master Database Mode	This option is only recommended for test environments. If two different Alfabet database s are used for configuring the solution and testing the application, the changes made to the database of the solution configuration environment can be uploaded to the database of the test environment via a direct connection between Alfabet Expand of the test environment with the Alfabet database of the solution configuration environment.
	For this scenario, the way updates are performed must be set with this attribute.
	Select **Complete Updates** if you want the complete configuration to be taken over. Select **Selective Updates** if the person triggering the updates shall decide about which parts of the configuration to take over.
	This option is only taken into account if the functionality is activated via the **Enable Master Database Configuration** attribute and a server alias is selected in the **Alias for Meta-Model Design Master Database** attribute. For a description of the complete configuration, see Configuring Alfabet Expand to Update the Configuration from a Master Database via a Direct Connection.
File-Based Updates Permitted	Select the checkbox to activate the **Update Meta-Model** functionality in Alfabet Expand Windows for update of the meta-model configuration of the current database with the meta-model configuration stored in an AMM file. This attribute is activated by default.
Enable Master Database Configuration	This option is only recommended for test environments. If two different Alfabet database s are used for configuring the solution and testing the application, the changes made to the database of the solution configuration environment can be uploaded to the database of the test environment via a direct connection between Alfabet Expand of the test environment with the Alfabet database of the solution configuration environment.
	This functionality is activated by selecting the checkbox of the **Enable Master Database Configuration** attribute.
	This option is only taken into account if a server alias is selected in the **Alias for Meta-Model Design Master Database** attribute. For a description of the complete configuration, see Configuring Alfabet Expand to Update the Configuration from a Master Database via a Direct Connection.

Parameter	Description
Scan uploaded files for malware	Select the checkbox to activate anti-virus check of Microsoft® Word® and PowerPointMicrosoft® PowerPoint® templates prior to uploading the template into the publication definition in the **Publications** explorer in Alfabet Expand. This option is activated by default.
Automatically generate ALFA_MM tables	Select the checkbox to activate update of the `ALFA_MM_*_INFO` object class tables at the end of each update of the meta-model via AMM file. By default the option is activated. The tables store the information about the current configuration of the Alfabet class model and should only be changed using this mechanism or the update option Meta-Model > Generate Meta-Model Information in the menu of Alfabet Expand. For more information about the `ALFA_MM_*_INFO` object classes, see *Getting An Overview of the Current Configuration* in the reference manual *Configuring Alfabet with Alfabet Expand*.
Automatically generate ALFA_PM tables	Select the checkbox to activate update of the `ALFA_PM_*_INFO` object class tables at the end of each update of the meta-model via AMM file. By default the option is deactivated. The tables store the information about the current configuration of the Alfabet presentation model and should only be changed using this mechanism or the update option **Meta-Model** > **Generate Presentation Model Information** in the menu of Alfabet Expand. For more information about the `ALFA_PM_*_INFO` object classes, see *Getting An Overview of the Current Configuration* in the reference manual *Configuring Alfabet with Alfabet Expand*.

Application Server tab

Use Event Queue for All Jobs	If **Use Event Queue for All Jobs** is selected, the new way to schedule server processes via event queueing is used. All other settings in the Application Server tab will be ignored because they refer to the old way to handle processes via remote processing. This is the recommended option that must be actively selected for existing server alias configurations to change to the new processing mode.
Use Application Server and Net Remoting Service	If **Use Application Server and Net Remoting Service** is selected, the old way of direct service calls between the Alfabet Web Application and the Alfabet Server is used. Please note that this option will be deprecated in the future.
Remote Alias for Connection to the Application Server	Only if **Use Application Server and Net Remoting Service** is selected: This attribute is required for the connection of the Alfabet Web Application to the Alfabet Server. This connection is mandatory for the following functionality: • using the **Data Capture** functionality, • performing data anonymization in the **Users Administration** functionality.

Parameter	Description
	Optionally, it can be used to execute the following to enhance performance: • Executing the full-text search. • Sending emails. • Migrating workflows. • Executing ADIF jobs. Select the remote alias used for connection to the Alfabet Server in the drop-down field. **Note:** The Alfabet Server and the Alfabet Web Application must connect to the same database, but with a different server alias. The remote alias must be defined for the server alias of the Alfabet Server and must be available in the AlfabetMS.xml file of the Alfabet Web Application in addition to the server alias of the Alfabet Web Application.
Use Server to Execute Full-Text Search	Only if **Use Application Server and Net Remoting Service** is selected: Select this checkbox to use the Alfabet Server instead of the Alfabet Web Application to execute the full-text search while using Alfabet.
Use Server to Update Work-flow Tem-plates	Only if **Use Application Server and Net Remoting Service** is selected: Select this checkbox to use the Alfabet Server instead of Alfabet Expand Web to execute work-flow migration.
User Server to Execute ADIF Jobs	Only if **Use Application Server and Net Remoting Service** is selected: Select this checkbox to use the Alfabet Server instead of the Alfabet Web Application to execute ADIF jobs triggered via the Alfabet user interface or a REST API call to the Alfabet Web Application.

Server Settings > **General** tab

Parameter	Description
Database Con-nection Count	This attribute is only used for connection between the Alfabet Server and the Alfabet database. The maximum number of parallel connections allowed from the Alfabet Server to the database. The default value is 50. This is the minimum allowed value. If the value is set to a value lower than 50, it will not be stored and the default is maintained. The maximum possible number of connections depends on the database installation and administration settings. The

Parameter	Description
Allow User Specific Configuration	Select the checkbox to allow each Alfabet user to configure the layout of standard Alfabet views displaying a tabular report individually. When set to false, configuration of the layout of tabular reports is restricted to a solution designer accessing the Alfabet solution via Alfabet Expand. The configuration is then used for all users.
	NOTE: Tabular configured reports are always configurable by the Alfabet user independent of the setting defined in the **Allow User Specific Configuration** attribute.
	NOTE: The setting must be identical in both the server alias for accesssing the Alfabet database with Alfabet Expand and the server alias for the Alfabet Web Application !
	NOTE: For information on the configuration of table layouts, see the section *Configuring a View Scheme for a User Profile* in the reference manual *Configuring Alfabet with Alfabet Expand*.
Allow Edit Object Colors	Select the checkbox to display the **Colors** button in the toolbar of Alfabet business graphics. When set to "`true`", the button will be available in all relevant standard and configured views. Any user regardless of object authorization can edit the color used to display an object in business graphics. The color will be used for the object in all business graphics in Alfabet and will be displayed for all users. If you do not select the checkbox, the **Colors** button will not be displayed in any standard or configured views in Alfabet.
Use Recipient's User Profile for External Links	This setting is only relevant, if the **External Access** attribute in the **Server Settings > Security** tab is set to **Allow As Authenticated User**. If you deselect the checkbox, links to the Alfabet user interface in Alfabet generated emails will open with the user profile of the user sending the mail. If you select the checkbox, the links will open the user interface with the user profile of the user receiving the email.
Send Web Message Direct	Select the checkbox if Web messages should be sent directly from Alfabet to the user creating the message associated with a page view. Otherwise, an email client will open. Selecting the checkbox is primarily applicable to enterprises using older email clients that cannot correctly process Alfabet-generated messages containing special characters.
Allow Usage of Run-Time Fonts	GUI schemes can be configured in Alfabet Expand to change the outlook of the Alfabet user interface. The default GUI scheme used for display of the Alfabet user interface uses the `Roboto` font as standard font for the user interface. The font is not a standard font available via the web browser, but needs to be uploaded from the Alfabet Web Application at runtime. Upload is only performed if the **Allow Usage of Run-Time Fonts** attribute is selected in the server alias settings of the Alfabet Web Application. If the attribute is not selected, the fonts defined for the **Application Font Fallback 1** and **Application Font Fallback 2** attributes of the GUI scheme will be displayed on the Alfabet user interface.

Parameter	Description
Use ReadOnly User for Report Execution	Select the checkbox if you want configured reports to be executed with a system generated user with ReadOnly access rights. This method prevent damage to the database caused by configured reports based on native SQL that contain `DELETE`, `DROP` or `UPDATE` statements. Please note that configured reports based on stored procedure will not work if this option is selected. This option will be ignored if configured reports are configured to be executed with customer defined database users with explicit access permissions. For information about the configuration required to access the Alfabet database with different access permissions per configured report, see Executing Configured Reports with a Different Database User.
Update History User Name	The attribute defines the storage mode of the user's name in the audit history of objects in Alfabet, the Windows event logging and the history of anonymization actions and updates of the meta-model from a master database. If set to `USER_NAME` the user's user name is used in the audit history table. If set to `TECH_NAME`, the technical name of the user is used. **NOTE**: If the attribute is set to `TECH_NAME`, a technical name should be defined for all Alfabet users. If no technical name is configured for a user, the user name of that user is stored in the audit tables.
Broadcast Messages	Select the checkbox to enable the sending of broadcast messages to all users logged in to the Alfabet user interface. For more information about broadcast messages, see the section *Defining Broadcast Messages for the User Community* reference manual *User and Solution Administration*.
Control Client Aliases	This attribute is only relevant when an Alfabet Server is used to run tools that require an indirect connection to the Alfabet database via the Alfabet Server. Set the checkbox to control the authentication settings of the remote alias configurations on server side. Selecting the **Control Client Aliases** attribute requires that all relevant remote alias configurations are available in the `AlfabetMS.xml` configuration file of the Alfabet Server. For more information, see Configuring the Alfabet Server to Control Client Authentication Settings in the section Configuring User Authentication.
Stacked Wizards	Select the checkbox to allow a wizard to be opened from within a wizard view displayed in the wizard that a user is currently working in. The new wizard window will overlay the wizard the user is originally working with. The user can only return to the original wizard, when the new wizard has been closed. For more information about the configuration of wizards, see *Configuring Wizards* in the reference manual *Configuring Alfabet with Alfabet Expand*.
Enable Audit	Select the checkbox to enable history tracking for the Alfabet Server. (To activate history tracking for the Alfabet Server, use the context menu of the respective server alias as described in the section Activating the History Tracking Functionality) When history tracking is activated, a database table named <name of database table for

Parameter	Description
	object class>_AU is created for each object class with the **Audit** attribute set to `true`. Whenever an object is changed, a new line with relevant information about the action performed is written to the audit table. **NOTE**: The history tracking functionality cannot be implemented for the object class `Person`.
Track User Login	Select the checkbox if you want to track user login to the Alfabet user interface. If the **Track User Login** attribute is activated, the user name and user profile used for login will be written to a database table together with information about the time of the login. This information can be used to configure reports that provide usage statistics.
Enable Collaboration	Select the checkbox if you want to activate the collaboration functionality. In the collaboration functionality, a user can start a conversation about an object and invite other users to contribute to the discussion. By default, collaboration is activated.
ADIF Job Server Sleep Time (milliseconds)	Enter the time between checking the queue for ADIF jobs to be executed. The default of 3000 milliseconds can be changed as needed.
Event Server Sleep Time (milliseconds)	Enter the time between checking the queue for events to be executed. The default of 500 milliseconds can be changed as needed.
Document Storage Type	Determines whether documents uploaded to the Alfabet interface as attachments are stored in the **Internal Document Selector** of Alfabet or in a local file system. If set to `IDoc` or `DefautIDocfolder`, documents will be uploaded to the **Internal Document Selector**. If set to `ExternalFileSystem`, documents will be uploaded to the location specified via the **Document Storage Path** attribute. for details about the available storage types, see [Making Documents and Files Available to the Alfabet User Community](#).
Document Storage Location	Only valid if the **Document Storage Path** attribute is set to `ExternalFileSystem`. Specifies the absolute path to the location in the local file system where documents uploaded to the Alfabet interface are stored. Click the Browse button to select a directory. **NOTE**: The Alfabet Web Application must have Read/Write access permissions to the selected directory. The directories are not protected by Alfabet access permissions but by the access permissions granted on the local network for the directories.

Parameter	Description
	Server variables can be used to define part of the information as a variable with the value set in the **Variables** tab of the server alias configuration. For more information, see Defining Connections Based On Server Variables.

Server Settings > **Security** tab

Parameter	Description
Ignore User Domain	By default, Alfabet user names for Windows sign-on include the Windows domain name when created in the format *Domain Name\User Name*. Select the checkbox to drop the domain name when creating the Alfabet user name. For every re-login by the user, the domain name will accordingly be included or excluded from the Windows network name for the user. This setting should only be used if the only user names are unique across the enterprise.
Allow Anonymous User	Select the checkbox to allow unrestricted access to Alfabet views from links in external applications or emails generated via Alfabet, like object and express view emails or notification emails. When the checkbox is not selected, only named users can access the view and access from links from express view emails is not allowed at all. Anonymous login is not possible using Windows Sign-On. Login as a named user via a login screen is required for standard login. For more information see Configuring the Setup To Open the Alfabet Interface via Links in Email Notifications.
External Access	Select how access permissions to Alfabet views opened via links in emails shall be handled. For more information, see Managing Access Permissions for Access from External Applications.
Allow Re-Login	This attribute is only required when single sign-on is selected for user authentication. If the checkbox is selected, the login screen that is displayed for re-login when a user logs out includes an option for re-login using standard login with a Alfabet specific user name and password. If the checkbox is not selected, the **Log out** option will open a login screen restricted to re-login via single sign-on only.
Maximum Number of Failed Logins	Enter the number of consecutive failed login attempts that are allowed for a user. If the user attempts to login and login fails because a wrong password has been entered for the configured number of attempts, the user will be blocked and any further login attempts will be rejected. The number of consecutive failed logins will be counted and set back to zero with every successful login. If a user is blocked, a user administrator must re-set the count in the **User Administration** functionality. This setting is only evaluated if the **Enable Event Logging** and **Enable Login and Logout Tracking** attributes in the **Server Settings** > **Logging** tab are activated. Setting the **Maximum Number of Failed Logins** attribute to-1 sets the maximum number of failed logins to unlimited.

Parameter	Description
Validate ADIF Parameters for SQL Injection	This setting is only evaluated for attribute settings in old ADIF schemes that have been defined prior to Alfabet 10.4 and have the **Parameters Backward Compatibility Mode** attribute set to `True`. If you are running such ADIF schemes, select the checkbox to check attribute values handed over during ADIF scheme execution for SQL injection vulnerability. The mechanism checks whether an attribute value is either a number or written in single quotes. If both condition do not apply, the ADIF scheme will not be executed and an error message will provide information about the incorrect attribute value definition.
Path for Self-Signed Public Certificate Files	For integration solutions based on web services (such as ARIS- Alfabet Interoperability Interface, import of data from Technopedia®, or Jira® integration), self-signed certificate validation can be used on HTTPS connections. The following configuration is required: • The self-signed certificates from the third-party web service must be copied to a local folder that the Alfabet Web Application has access permissions to. • The path to the folder must be defined in the **Path for Self-Signed Public Certificate Files** attribute.
Enable Changes to Own Authorization	By default, user administrators do not have permission to edit their own data in the **Users Administration** functionality. Selecting the checkbox will give user administrators permission to edit their own data.

Server Settings > **Logging** tab

Parameter	Description
Activate Logging	Select the checkbox to configure the Alfabet component to write log messages to the Windows Event Log and/or to activate writing of log messages into a central log file or an external Seq® server.. For more information about writing log messages to the Windows Event Log, see the section Tracking Login Actions in the Windows Event Log. For more information about implementing central logging, see Logging of Alfabet Functionality.
Windows Event Logging	The attribute **Event Source Name** displays the alias name of the Alfabet component by default. The name defined with this attribute is displayed in the Windows Event Log as log source if **Activate Logging** is activated. The setting requires that that **Enable Login and Logout Tracking** attribute is also activated after the server alias has been configured, that the Alfabet Administrator is run as administrator, and that the **Register Event Logger** option in the context menu of the server alias is triggered. For more information, see the section Tracking Login Actions in the Windows Event Log.

Parameter	Description
File Logging	If log information shall be written to a central log file, attributes in this section must be defined: • **Log File Directory**: Enter the absolute path of the log file directory or select a directory from the local file system using the Browse ![button] button. Both the log files and the archive ZIP files for the log files will be written into the selected directory. Server variables can be used to define part of the information as a variable with the value set in the **Variables** tab of the server alias configuration. For more information, see [Defining Connections Based On Server Variables](#). • **Log File Max Part Size MByte**: Define the maximum allowed file size in megabyte. The log file will be archived and a new log file started if either the defined file size has been reached or the time defined with the **Log Rotation Max. Time (Hours)** attribute has elapsed. • **Log Rotation File Name Suffix**: Select GUID to add a GUID to the ZIP file and the log file in the ZIP file on archiving of the log file. • **Log Rotation Max Time (Hours)**: Define the maximum time between archiving of the log file in hours. The log file will be archived and a new log file started if either the defined time has elapsed or the file size defined with the new **Log File Max Part Size MByte** has been reached. The setting requires that the **Tracking Log Level Options** content is set and that **Activate Logging** is activated.
External Logging	If the log information shall be send to a Seq server, the attributes in this section must be defined: • **Server Type**: Currently only Seq can be selected. • **Server URL**: The URL of the HTTP API of the Seq server. • **Access Key**: The access key for access to the HTTP API of the Seq server. • **Environment**: If multiple instances are run on the same host for production, validation and development and all instances are configured to send log messages to the same Seq server, the type of environment can be selected from the drop-down list. The selected environment type will be included in the messages to allow the different environments to be distinguished. These settings requires that the **Tracking Log Level Options** content is set and that **Activate Logging** is activated.
Server Settings > **Logging Details** tab	If information shall be written into a central log file or send to a Seq® server or login actions shall be logged into the Windows event log, the log level must be defined for the respective activity in this field.

Parameter	Description
	• `UserLogon`: This option is not relevant for central logging but for writing information about login and logout activities into the Windows event log. Click in the cell in the **Event** column to activate the row. For more information, see Tracking Login Actions in the Windows Event Log.
	• **System**, **Workflow**, **ADIF**, **Report**, **Conditions**, **REST**, **SSO**, **License**, **MonitoringEvents**, **WebApplication**,**GitIntegration**: Activates central logging of the respective functionalities with the log level defined in the column **Level**. to activate logging in a central log file, click in the respective cell in the **File** column. To activate sending of log information to a Seq server, click in the respective cell in the **External** column. For details about the information written into the log file for each option, see Central Logging of Functionality for Alfabet Components.
	• **MonitoringEvents**: Activates logging for the monitoring of database availability. This option must be activated to test the database availability via the mechanism described in the section Checking the Accessibility of the Alfabet Database.

Server Settings > Tracking tab

Track Usage	Select the checkbox to log the usage of the Alfabet functionalities by user profiles used to access the functionalities. You must track usage if you use Alfabet with a metered contract. The Alfabet Server writes information about the usage of modules to a `DAT` that is stored in the directory specified with the **Track Usage Directory** attribute.
Track Usage Directory	Select a directory to store the log files for the track usage functionality of the Alfabet Server, which is activated by means of the **Track Usage** attribute. By default, the files are stored in the working directory of the Alfabet Server. Server variables can be used to define part of the information as a variable with the value set in the **Variables** tab of the server alias configuration. For more information, see Defining Connections Based On Server Variables.
Track Presentation Usage	A presentation usage tracking mechanism can optionally be activated that tracks all user activity in the Alfabet user interface. This mechanism can help to understand which functionalities are most used and which functionalities are obsolete. Select **Local Database** to activate presentation usage tracking or select **Deactivated** if you do not want user activity to be tracked. The **Alfabet REST Service** option is for future use only and cannot currently be selected.
Track User ID	If presentation usage tracking is activated and this checkbox is selected, the REFSTR of the user will be tracked in addition to the session ID of the current user session and sub-session. If the checkbox is not selected, only the session ID and sub-session ID of the current user session will be saved in the tracking records. This information is sufficient to evaluate which views have been accessed by a user during the same session without adding actual user information to the tracking information. It is

Parameter	Description
	recommended that you check the requirements regarding legal compliance of tracking user information prior to selecting the **Track User ID** checkbox. Please note that the functionalities **List of visited objects**, **List of visited reports**, and **Recommended Navigations** require that presentation usage tracking is activated with activated tracking of the user ID.
Alfabet REST Service Connection Name	This attribute is currently not used.
Archive/Restore Presentation Usage Tracking	If you select the checkbox, the presentation tracking information will be included in Alfabet Database Archive files (ADBZ files) and restored in target databases when restore is performed from an ADBZ file. The presentation usage options as well as this attribute must be activated on both the alias used for archiving the database and the alias for restoring the database to include the information.
REST API tab	
Enable REST API v1	Select the checkbox to enable the Alfabet interface for RESTful Web Services, version 1. This option must be enabled to activate the ARIS- Alfabet Interoperability Interface. For general use of the Alfabet REST API for access to the Alfabet database with external applications **Enable REST API v2** attribute must be enabled.
Enable REST API v2	Select the checkbox to enable sending RESTful Web Services requests to the Alfabet Web Application from external applications.
API Access Options	Select the checkbox of each RESTful service endpoint that shall be accessible. For detailed information about the access conditions required for different functionalities see *Activating the Alfabet RESTful API on Server Side* in the reference manual *Alfabet RESTful API*.
Server Settings > **Email Settings** tab	
EMail Mode	Select `SMTP` to send emails via an SMTP server. The option `MS Graph API` is Alfabet is only available for future use.
SMTP Server	Enter the fully qualified domain name of the SMTP server that is used to send email messages (for example, for Alfabet assignments, notifications, or express views) from the Alfabet Web Application or the Alfabet Server to the relevant Alfabet users.

Parameter	Description
System Sender Email Account	When the **System Sender Email Account** attribute is set, the Alfabet Web Application or the Alfabet Server use the specified email address as sender address for all emails that are automatically sent from the Alfabet Web Application or the Alfabet Server even if it is possible to specify the user that triggered the sending of the email.
	When the **System Sender Email Account** attribute is not set, the Alfabet Web Application or the Alfabet Server read the email address of the sender from the user configuration for the user that is the originator of the email. When the originator is not specified or a batch job is run, the sender email address is a non-existent system email address.
	NOTE: You must enter an existing valid email address for the **System Sender Email Account** attribute.
Failover Sender Email Account	This attribute is only relevant when the **System Sender Email Account** attribute is not set. The Alfabet Web Application or the Alfabet Server then use the email address defined in the Alfabet user configuration for the originator as sender email address for outgoing emails.
	When **Failover Sender Email Account** attribute is set, the Alfabet Web Application or the Alfabet Server use the specified email address as sender address if the email address of the originator is not defined.
	When **Failover Sender Email Account** attribute is not set, sending of emails fails if the email address of the originator is not defined.
	NOTE: You must enter an existing valid email address for the **Failover Sender Email Account** attribute.
Test Receiver Email Account	Specify a valid email address to send all emails that are generated by Alfabet to the email address. The email configuration of the Alfabet user that is the recipient of the email is ignored.
	If the field is empty, emails are sent to the email address defined for the recipients of the emails in the user configuration of Alfabet.
Send Email Timeout (milliseconds)	The timeout in milliseconds for sending emails via the Alfabet Web Application or the Alfabet Server. The Alfabet Web Application or the Alfabet Server wait the defined time for the SMTP server to send out outgoing emails. If the timeout elapses before the emails are be sent, sending of emails will fail.
Max. Emails per Minute	Sending of emails is done asynchronously. Messages are queued and the sending of emails is performed in a separate process independent of the user session. This attribute defines the maximum number of emails processed per minute. The default is 30. The attribute can be changed according to the setup of the SMTP server.
Email Queue Sleep Time (milliseconds)	The Alfabet Server will check the queue for emails to be sent out in the specified time interval. The default value is 3000.

Parameter	Description
Azure Application	This field is for future use only and only visible if the **EMail Mode** attribute is set to `Microsoft Graph API`.
Proxy	This field is for future use only and only visible if the **EMail Mode** attribute is set to `Microsoft Graph API`.

Client Settings > **General** tab

Save Recent Objects	Select the checkbox if the recent objects data is saved when ending a Alfabet session.

Client Settings > **Authentication** tab

Mode	In the drop-down field, select the authentication method used to check user credentials: • `Standard`: Standard login with user name and password. • `ExternalSource`: User authentication via an external LDAP server. For more information about using an external data source for user authentication, see the section Configuring the Alfabet Web Application to Perform User Authentication Based On User Data from an External LDAP Data Source in the section Integrating Data from External Sources. The **External Source** attribute must be defined if `ExternalSource` is selected. • `SSO_WinUser`: Windows Sign-On is used for user authentication. The Windows login data are used to check user credentials. This method can only be used for access to the Alfabet user interface via the Alfabet Web Application. • `SSO_Portal`: Access to Alfabet is granted to all users successfully logged in to a company authentication portal. This method can only be used for access to the Alfabet user interface via the Alfabet Web Application. • `SSO_Certificates`: Access to Alfabet is granted to all users authenticated via client server Web certificates. This method can only be used for access to the Alfabet user interface via the Alfabet Web Application. • `SSO_FederatedAuthentication`: Access to Alfabet is granted to all users successfully logged in to a company's federated authentication system. This method can only be used for access to the Alfabet user interface via the Alfabet Web Application. • `SSO_CustomMethod`: User authentication is exclusively performed via execution of a custom code.

Parameter	Description
	For more information about user authentication, see the section <u>Configuring User Authentication</u>.
Request Credentials URL	A link **Click here to request access credentials** can be displayed on the login screen. New users that would like to have access to Alfabet but do not yet have a user name and password assigned can click the link that will lead them either to a URL for web-based request of user credentials or will open an email to a predefined email addressed to the system administrator granting access to Alfabet.
	The link for request of access credentials will only be displayed if this attribute is set.
	If a URL is defined in the field, the link on the login screen will open the defined URL. The URL must be defined starting with http:// or https://.
	If an email address is defined in the field, the link in the login screen will open an email to the defined email address with the default mail client of the user with the subject line **Access Credential Request**. The email address must be defined as `mailto:` followed by the email address.
HTTP Server Variable for Portal Integration	If the **Mode** attribute is set to `SSO_Portal`, enter the name of the server variable corresponding to the HTTP header that contains the user name.
Certificate Attribute	If the **Mode** attribute is set to `SSO_Certificates`, enter the name of the certificate attribute that is used to identify the user. Depending on the setting of the **Certificate Value Format** attribute, either the whole attribute is used as user name for the authentication, or the attribute is scanned for text written in parenthesis and the text in parenthesis is then used as the user name. The user name may be optionally amended by a specified prefix or suffix defined via the **User Name Prefix** and **User Name Suffix** attributes.
Certificate Value Format	If the **Mode** attribute is set to `SSO_Certificates`, select the method used to read the user name from the certificate attribute defined in the **Certificate Attribute** attribute.
	If `EntireValue` is selected, the entire value defined in the **Certificate Attribute** attribute will be used as user name for the authentication. If `Parentheses` is selected, the attribute will be scanned for text written in parenthesis and the text in parenthesis is then used as the user name. The user name may be optionally amended by a specified prefix or suffix defined via the **User Name Prefix** and **User Name Suffix** attributes.
User Name Prefix	If the **Mode** attribute is set to `SSO_Certificates` and the **Certificate Value Format** attribute is set to `Parentheses`, enter a prefix that shall be added to the certificate attribute part in parentheses to generate the login name for the user, if applicable.

Parameter	Description
User Name Suffix	If the **Mode** attribute is set to `SSO_Certificates` and the **Certificate Value Format** attribute is set to `Parentheses`, enter a suffix that will be added to the certificate attribute part in parentheses to generate the login name for the user, if applicable.
Authentication Connection Test Log File	For all Single Sign-on authentication mechanisms, the information about the authentication process can be written to a log file. Optionally, you can change the name of the authentication log file. If you do not specify a path, the file will be located in the physical directory of the Alfabet Web Application. The path specification must be an absolute path. This file is only relevant for a first test of the connectivity. During normal operation, the field should be cleared. Make sure that the Alfabet Web Application has Write permissions for the file.
External Source	If the **Mode** attribute is set to `ExternalSource`, enter the name of the external source or external source pool that shall be used for user authentication. The name is specified with the **Name** attribute of the external source (pool) in the external source configuration. For more information about the configuration of the external source, see [Configuring Access to an External Data Source and Mapping of Data](#).
Custom Method	Client-side hook for additional customer needs regarding a credential check of Alfabet users. Contact Software AG Support for further information.

Client Settings > **Actualization** tab

Mode	If user authentication is performed via an LDAP external source, and multiple `ID`s are available for one `distinguished name` (`DN` name), the Alfabet components can be configured to store the set of ID values in a custom attribute defined by the customer for the class Person of the Alfabet meta-model. Select `ExternalSource` to activate actualization of the compound user ID of the user during login with the `ID`s from the external source or `None` to deactivate actualization. For more information about actualization, see the section [Configuring the Alfabet Web Application to Perform User Authorization Based On User, User Group and User Profile Data from an External Data Source](#).
Compound User ID	Enter the name of the custom property for the object class `Person` that should be used to store the compound user ID.
External Source	Enter the name of the external source or external source pool that shall be used for actualization of the data in the compound user ID property. The name is specified with the **Name** attribute of the external source (pool) in the external source configuration. For more information about the configuration of the external source, see [Integrating Data from External Sources](#).

Parameter	Description
Custom Method	Client-side hook for additional customer needs regarding a credential check of Alfabet users. Contact Software AG Support for further information.

Client Settings > Authorization tab

Parameter	Description
XML Object	Enter the name of the XML object containing the authorization configuration in the **XML Object** field. The name is either `AuthConfiguration` or the name defined with the **Name** attribute of the copy of the XML object ***AuthConfiguration***.
Custom Method	Client-side hook for additional customer needs regarding a credential check of Alfabet users. Contact Software AG Support for further information.

User Password Settings tab

Parameter	Description
Interval	Number of days that a password is valid. After the defined time ends, the user is prompted to select a new password. Enter "-1" to configure that the user will never be prompted to change the password.
Enforce Password Criteria	Select the checkbox to use the password security options defined below. User passwords that do not correspond to the configured attributes are invalid. • **Minimum Password Length**: Enter the minimum number of characters required for a user password. • **Recent Passwords Number**: Enter the number of passwords that will be saved for the user and thus cannot be reused. • **Min. Lowercase Letters**: Minimum number of lower case letters required in a password. • **Min. Uppercase Letters**: Minimum number of upper case letters required in a password. • **Min. Digits**: Minimum number of digits required in a password. • **Min. Special Characters**: Minimum number of special characters required in a password.
Reset/Regenerate Password Log File	Enter the absolute path to and name of the log file that shall be used to log any password reset and password regeneration action. By default, the log file is stored in the directory `/AlfaWebAppplication/Runtime/Tmp` of the Alfabet Web Application with the name `PasswordChangeLog.txt`.

Parameter	Description
Enable Forgot Password	Select the checkbox to display a **Forgot your password?** link to the user on the login screen. If the user clicks the link, the **Regenerate Password** functionality that assigns a new password to the user via email will be automatically invoked. The user will be requested to change the password at first login with the automatically generated password. This functionality requires that the sending of emails is correctly implemented for the Alfabet Web Application.
Maximum Number of Regenerated Passwords	Each time a password is regenerated via email, either by a user administrator in the **Users Administration** functionality or because the user clicks the **I forgot my password** link on the login screen, a counter is incremented. The counter is reset to zero on login of the user with the last regenerated password. If the counter reaches a configured maximum number, the password regeneration functionalities are deactivated for the user unless the counter is reset via the **Action** > **Reset Regenerated Passwords Counter** option.

Database Settings > **Details** tab

Parameter	Description
Database Driver	Select `SqlServer` if the database type is a Microsoft® SQL® server database or `Oracle` for an Oracle® database.
Database Driver Type	If the **Database Driver** attribute is set to `SQL Server`, select one of the following: • `Microsoft.Net Framework` for the driver supporting.NET Framework only. This driver has been used for all installations prior to release 10.9.0 as only supported driver. • `Microsoft Sql Client` for the driver supporting both.NET Framework and.NET Core. This is the default setting for new installations.
User Name	Enter the user name for the connection to the database. The user name is stored encrypted in the `AlfabetMS.xml` configuration file.
Password	Enter the password for the connection to the database. The password is stored encrypted in the `AlfabetMS.xml` configuration file.
Database	Enter the name of the Alfabet database: • For a Microsoft® SQL Server® database, use the notation servername\databasename for a default instance or servername\instancename\databasename for a named instance. • For an indirect connection to an Oracle® database, use the Net Service Name of the database configured with the Oracle® Net Assistant. • For a direct connection to an Oracle® database this attribute is ignored.

Parameter	Description
Encrypt Connection	For a connection to a Microsoft® SQL Server® only: Select the checkbox to apply SSL to the connections between the Alfabet component and the Alfabet database.
Indirect Connection/Direct Connection	For a connection to an Oracle® database only: Select whether a direct or indirect connection to the Oracle® database should be established. The direct connection is recommended for connections to the Alfabet database for performance reasons. Indirect connections require additional installation of a client application. Some Oracle® features are only supported for indirect connections, but none of these features is required for working with Alfabet. It is possible to change to an indirect connection if any of the restrictions for direct connections become critical in the future. The main restrictions of direct connections to Oracle® databases are: • Only the TCP/IP protocol is supported for the connection. • Advanced authentication such as OS authentication and proxy authentication are not supported. • Advanced configurable Oracle® functionalities such as Real Application Cluster, Oracle Loader, Transparent Application Failover, and Oracle Transaction Guard are not supported.
Database Port	For a direct connection to an Oracle® database only. Enter the port of the database server host the Alfabet Server should use for connection to the Alfabet database.
Database Host	For a direct connection to an Oracle® database only. Enter the host name or IP address of the database server host.
Database Service Name	For a direct connection to an Oracle® database only. Enter the SID of the Alfabet database.
Rebuild fragmented indices on server start	This parameter is evaluated in the server alias configuration of the Alfabet Server only: Select the checkbox to activate rebuilding of fragmented indices on each start of the Alfabet Server. Adding data to and deleting data from database tables leads to database index fragmentation which has a negative impact on performance.
Clean invalid characters from strings during reading and writing	Select the checkbox if you want strings to be checked for validity prior to check-in or when reading the string from the Alfabet database. Invalid characters such as special characters that are not allowed in the string are cleared from the string and the string is written to the database and displayed in the Alfabet user interface without the special characters. ⓘ This option can cause a loss of performance. The checkbox should only be selected if problems with special characters have been detected.

Parameter	Description
Database Setttings > Command Details tab	
Default Execution Timeout (seconds)	Enter the timeout in seconds that shall be used as the default timeout for all processes targeting the database. If the field is empty, the default timeout of the database server is used.
User Interface Execution Timeout (seconds)	Enter the timeout in seconds for user activity via the Alfabet user interface except for the specific actions defined in the other timeout options. If the field is empty, the default timeout defined with the **Default Execution Timeout (seconds)** attribute is used.
Report Execution Timeout (seconds)	Enter the timeout in seconds for the execution of configured reports that are not configured to be executed offline. If the field is empty, the default timeout defined with the **Default Execution Timeout (seconds)** attribute is used.
	Please note that tabular configured reports with filters can be configured to be executed offline if a long execution time is expected. The report execution timeout defined in the server alias of the Alfabet Web Application is ignored for offline executed report. An individual maximum execution time is configured for each offline executed report. For more information about offline execution of configured reports, see in the reference manual *Configuring Alfabet with Alfabet Expand*
ADIF Execution Timeout (seconds)	Enter the timeout in seconds for the execution of ADIF jobs. If the field is empty, the default timeout defined with the **Default Execution Timeout (seconds)** attribute is used.
REST Execution Timeout (seconds)	Enter the timeout in seconds for execution of RESTful service call to the Alfabet RESTful services. If the field is empty, the default timeout defined with the **Default Execution Timeout (seconds)** attribute is used.
Expand/Administrator Execution Timeout (seconds)	Enter the timeout in seconds for user activity via the tools Alfabet Administrator, Alfabet Expand Web and Alfabet Expand Windows. If the field is empty, the default timeout defined with the **Default Execution Timeout (seconds)** attribute is used.
Variables tab	The **Variables** tab allows you to define server variables that can be used in Alfabet Expand for the definition of: • Connection strings to external data sources.

Parameter	Description
	• URLs defined for dynamic Web links (for more information see the section *Configuring Dynamic Web Links That Users Can Access* in the reference manual *Configuring Alfabet with Alfabet Expand*). • URLs defined for external reports (for more information see the section *Configuring Reports* in the reference manual *Configuring Alfabet with Alfabet Expand*). • URLs defined for custom online help (for more information see the section *Providing Custom Online Help to the User Community* in the reference manual *Configuring Alfabet with Alfabet Expand*). • Value definitions for XML attributes defining connections to external systems in the XML objects for the configuration of the interfaces to external systems. This applies to all XML objects located in Alfabet Expand in the **Presentation** tab in the **Integration Solutions** subfolder of the **XML Objects** folder. For more information, see the documentation of the respective interface. Use the **New** button to define new variables. A variable consists of a name and a value. It is stored encrypted in the server alias configuration. **NOTE**: The variable name can only contain letters (of the English alphabet), numbers and underscore. The following characters are not allowed in server variable values: " < > These characters can be written as HTML code in the server variable value: • `>` for > • `<` for < • `"` for "

License tab

Parameter	Description
Key	If you have licensed the Guide Pages Designer, the license key will be automatically added to the **Licenses** tab upon opening the server alias editor and stored when closing the editor via the **OK** button. If you have received a different license key from Software AG, you can overwrite the automatically set license key.
Summary	This tab provides information about the functionality that is included in the license key entered in the field of the **Key** tab.

Defining Connections Based On Server Variables

The definition of server variables allows you to store information about connection strings in the server alias configuration. Storing the information about the connection strings in the server alias configuration instead of directly defining them in the configuration enhances security. Server variables are stored encrypted in the server alias and a command line tool is available to allow server variables to be set without direct access to the server alias configuration. The administrator of the component the connection is configured to can set the server variables via the command line tool without being provided access to the complete server alias configuration and without providing the connection information to the administrator of the Alfabet components. For information about setting the server variables via the command line application, see Changing the Server Variables in the Server Alias Configuration.

Storing connection data in server variables also eases the propagation of changes.

> For example, if the setup of a connection to an external data source is done first in a test environment, the test environment is an exact copy of the production environment except that the components are installed on different servers. Therefore, all connections defined will be identical in the test and the production environment except for the server name. When migrating to the production environment, the server name must be changed in all configurations done in Alfabet Expand. But if the server name is defined as a server variable in the server alias configuration, the configurations carried out in the configuration tool Alfabet Expand will reference the server name as a variable in the connection string and can be reused in the production environment. Only the variable definition in the server alias configuration of the production environment must be set to the current value.

Server variables can be used in the following configurations:

- URLs and storage location definitions in the server alias configuration of the Alfabet components (For more information, see Configuration Attributes for the Alfabet Components).

- Connection strings defined in XML objects for the definition of interfaces with third party components. This applies to all XML objects located in Alfabet Expand in the **Presentation** tab in the **Integration Solutions** subfolder of the **XML Objects** folder. For more information, see the documentation of the respective interface in the reference manual *Configuring Alfabet with Alfabet Expand*.

- Connection strings to external data sources (for more information see Integrating Data from External Sources).

- URLs defined for dynamic Web links (for more information see the section *Configuring Dynamic Web Links That Users Can Access* in the reference manual *Configuring Alfabet with Alfabet Expand*).

- URLs defined for external reports (for more information see the section *Configuring Reports* in the reference manual *Configuring Alfabet with Alfabet Expand*).

- URLs defined for custom online help (for more information see the section *Providing Custom Online Help to the User Community* in the reference manual *Configuring Alfabet with Alfabet Expand*).

- Definition of database users with restricted access permissions in the XML object *DatabaseUsers* in the **Administration** subfolder of the **XML Objects** folder (for more information, see Executing Configured Reports with a Different Database User).

Either all or part of the connection string can be defined in a server variable. It is also possible to build the connection string from a number of concatenated server variables.

If an attribute or XML attribute in the above-mentioned configurations can be defined with server variables, this is explicitly mentioned in the documentation. If the use of server variables is not documented for an attribute, server variables cannot be used.

For example, a Microsoft® SQL server is used as an external data source. The server name and the database name should be defined as server variables because they are the subject of the change. Therefore, the following server variables are defined in the server alias configuration:

- SQLSERVER, specifying the Microsoft® SQL server used

- DBNAME, specifying the name of the external database

The connection string contains the variables instead of the current Microsoft® SQL server name and the database name:

```
ConnectionString="Data Source=$SQLSERVER;Initial
Catalog=$DBNAME;Pooling=false;Connection Reset=false;User
ID=Alfabet;Password=secret"
```

Server variables can be defined directly in the server alias configuration using the Alfabet Administrator:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer of the Alfabet Administrator.

2) In the table on the right, select the server alias for which you want to define a server variable and click the **Edit** button. The alias editor opens.

3) Go to the **Variables** tab and click the **New** button. A dialog box opens.

4) In the **Variable Name** field, enter a unique name for the server variable.

The server variable name may only contain letters (English alphabet), numbers, and the underscore symbol.

5) In the **Variable Value** field, enter all or part of the connection string used to connect to the external source.

6) Click **OK** to save your changes. The server variable definition appears in the list of server variables.

To edit or delete the server variable, select the server variable in the table and click the **Edit** or **Delete** button below the table.

7) Click **OK** to save your changes and exit the editor. The variable definition is stored encrypted in the server alias configuration and can be used in external source configurations.

In the definition of the connection string, a variable is included as $<server variable name>. For example, a server variable called SQLSERVER is referenced as $SQLSERVER. The variable definition can either substitute the entire connection string or a part of the connection string.

Deleting an Alias Configuration

⚠️   If you delete an alias, the configuration will be deleted irrevocably from the `AlfabetMS.xml` configuration file.

To delete a server alias or a remote alias configuration:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer. The right pane displays a list of all available alias configurations with a toolbar on top.

2) In the table, select the alias configuration that you want to delete.

    ℹ️   You can select several objects in the table at the same time by holding down the CTRL key while selecting.

3) In the toolbar, click the **Delete** 🗑 button.

4) Confirm the warning by clicking **Yes** or click **No** to exit without deleting the selected object(s).

Copying Existing Alias Configurations to a Separate AlfabetMS.xml File in Another Directory

Alias configurations are exclusively configured via the Alfabet Administrator. It is recommended that you create and maintain all alias configurations in the `AlfabetMS.xml` file located in the directory of the Alfabet Administrator, even if your installation is distributed to a multiple working directories or hosts, each containing a subset of the Alfabet components. After having created all required server and remote alias configurations for all Alfabet components, The **Distribute** functionality of the Alfabet Administrator can be used to create new `AlfabetMS.xml` files containing all or a subset of the alias configuration in the `AlfabetMS.xml` file of the Alfabet Administrator.

To create an `AlfabetMS.xml` file for distribution to other locations:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer. The right pane displays a list of all available alias configurations with a toolbar on top.

2) In the list, select all alias configurations that you want to include in the new `AlfabetMS.xml` file. You can use hold and click to select multiple alias configurations.

    ℹ️   If you create an `AlfabetMS.xml` file for the Alfabet Web Application and the is configured to connect to an Alfabet Server, the `AlfabetMS.xml` file must include both the server alias for connection of the Alfabet Web Application to the Alfabet database and the remote alias for the connection of the Alfabet Web Application to the Alfabet Server.

3) In the toolbar, select **Tools** > **Distribute...**.

4) In the dialog box that opens, choose a destination folder for the new `AlfabetMS.xml` file. The file name or folder must differ from the name and location of the `AlfabetMS.xml` file of the Alfabet Administrator.

5) Click **Save** to save the file.

Exporting a Report about an Alias Configuration

When you click an existing alias configuration in the explorer of the Alfabet Administrator, the right pane displays an overview report of the configuration for that alias. For security reasons, the password for connection to the database server is not included in the overview.

You can export this report as an HTML file or Microsoft® Excel® file.

To export the report:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer. The right pane displays a list of all available alias configurations with a toolbar on top.

2) In the table, select the alias configuration that you want to export the report for.

3) In the toolbar, click the **Export** button and select the file format for the report. You can export the data in the table to an HTML file or to a Microsoft® Excel® file.

4) In the dialog box that opens, select a location and enter a file name for the report and click **Save** to save the report.

Sorting the Alias Nodes in the Explorer Tree

By default, the alias configurations are listed in the explorer tree in the order of their creation. It is possible to sort the alias explorer nodes either alphabetically or in a defined order. The configured sort order is used in the explorer as well as the table displayed in the right pane when you click the node of the ascendant alias in the **Alfabet Aliases** explorer.

To sort the alias explorer nodes:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer. The right pane displays a list of all available alias configurations with a toolbar on top.

2) In the toolbar, select **Tools** > **Sort...**.

3) In the dialog box that opens, do either of the following:

   - To sort the alias explorer nodes alphabetically, select the **Sorted** checkbox at the lower left of the dialog box

   - To sort the alias explorer nodes in a customized order, select an alias from the list in the dialog box and use the up and down buttons in the dialog box to change the position of the alias in the list.

4) Click **OK** to save your changes.

5) Click **OK** to confirm the information in the message box.

Using the Context Menu Available via the Alias Explorer Node

The following table lists the functionalities that are available when you right-click a server alias in the explorer.

For some functionalities it is necessary to establish a connection to the Alfabet database by first selecting **Connect...** in the context menu. The table also specifies whether a connection is required to use a functionality.

If an action modifies the database content or the `AlfabetMS.xml` configuration file, this is also specified in the table.

When you select a functionality from the context that establishes a connection to the Alfabet database, a login window pops up. You must enter the user name and password defined for login to the Alfabet database.

When Windows sign-on is configured for connections to the Alfabet database, click **OK** without entering a user name and password. The Alfabet Administrator will connect to the Alfabet database via Windows sign-on and perform the selected operation.

Please note that on Oracle database servers, databases are stored under the login user. A database with the same name can exist under different user names. Login with a different user name than specified in the server alias you are currently working with will apply the changes to the database stored under the login user name rather than changing the database under the user name defined in the alias configuration.

The most important functionalities are listed in the following table. Only a short description of the functionality is provided with a link to the section of the reference manual *System Administration* that describes the functionality in detail.

Select...	In Order To....	Database connection required
Connect	Establish a connection to the database that is specified in the server alias configuration.	This action establishes a connection to the Alfabet database. Login to the database is required.
Disconnect	Close an existing connection to the Alfabet database configured in the server alias.	Yes, via the **Connect** functionality in the context menu.
Create the Alias as Copy	Create a new server alias configuration with the same attribute settings as the current server alias configuration. The attribute settings of the new server alias can then be modified as needed. For more information see the section Creating a Server Alias as Copy of an Existing Server Alias.	No

Select...	In Order To....	Database connection required
Create Remote Alias	Create a basic remote alias configuration for the selected server alias. The new remote alias appears as a new node in the explorer and the configuration attributes of the server alias (which are also required for the remote alias configuration) are automatically set. For more information, see the section Creating a Remote Alias in this chapter. **NOTE**: the new remote alias is directly written to the `AlfabetMS.xml` configuration file.	No
Archive Current Database	Create an ADBZ archive file of the Alfabet database that this server alias connects to. **NOTE**: A **Squeeze Audit Tables** option is available that scans the audit tables prior to archiving the database. Any entries that were generated during, for example, a batch update of data via batch utilities where no audit-relevant changes were documented will be deleted from the audit tables prior to archiving the database. The **Include User Settings** option allows user settings to be added to the archive that are defined for the current database.	Yes, via the **Connect** functionality in the context menu.
Restore Database Archive	Restore an ADBZ archive or an ADB archive file of an Alfabet database in the database that this server alias connects to. Please note the following: • This action overwrites the content of the Alfabet database. **Never restore a database without prior backup of the current database! If the restore action fails, it may result in a corrupt database!** • The restore of a database will fail if the server alias is configured to enforce password criteria and the database in the ADBZ file contains users with empty passwords or with a password not matching the defined user password criteria. If restore of the database fails, the target database may be corrupt. • This mechanism is only available if the database server is installed on the same machine as the Alfabet Administrator. It is not intended for use in productive environments and should only be used to ease database maintenance in development or test environments. • **Ignore case sensitivity validation**: A case-sensitivity check has been implemented for restore of the Alfabet database from ADBZ file. If the target database is case-insensitive and the ADBZ file was created from a case-sensitive database, the restore process will not be performed and a message informs about the	This action establishes a connection to the Alfabet database. Login to the database is required.

Select...	In Order To....	Database connection required
	incompatibility of the databases. Select the checkbox to deactivate the check and to allow a case-sensitive database to be restored to a case-insensitive target database. The attribute is deselected by default. • Prior to using this functionality, ensure that no Alfabet components are currently connected to the Alfabet database. This includes the Alfabet Web Application, the Alfabet Server (service), Alfabet Expand or batch utilities. A **Squeeze Audit Tables** option is available that scans the audit tables for obsolete entries. Any entries that were generated during, for example, a batch update of data via batch utilities where no audit-relevant changes were documented will be deleted from the audit tables restored from the ADBZ file.	
Release Restricted Mode	During update of the meta-model and during restore of the database, the Alfabet database is set to a restricted mode and connections to other Alfabet components are closed and new connection cannot be re-established. If the restricted mode is not automatically released after the change to the Alfabet database has been performed, the restricted mode can be released by using this option.	No
Activate Audit	Activates the history tracking for the Alfabet Web Application connecting to the Alfabet database with this server alias. If history tracking is activated, changes to objects will be documented in audit tables in the Alfabet database for all object classes that are configured to be audited. For more information about configuring and activating the history tracking functionality, see Activating the History Tracking Functionality.	Yes, via the **Connect** functionality in the context menu.
Deactivate Audit	Deactivates the history tracking for the Alfabet Web Application connecting to the Alfabet database with this server alias. If history tracking is deactivated, changes to objects will not be documented in the audit tables in the Alfabet database. For more information about configuring and activating the history tracking functionality, see Activating the History Tracking Functionality.	Yes, via the **Connect** functionality in the context menu.
Execute Script	Execute a script delivered from Software AG Support. **NOTE**: This action can modify the content of the Alfabet database.	Yes, via the **Connect** functionality in the context menu.
Update Meta-Model	Replace or merge the customer configuration of the solution environment of the current database with the customer configuration	This action establishes a connection to the Alfabet database. Login

Select...	In Order To....	Database connection required
	saved to an *.amm update file. For more information, see Administration Tasks Related to Solution Design. ⚠ Always back up the target database prior to performing an update of the meta-model. **NOTE**: Prior to using this functionality, ensure that no Alfabet components are currently connected to the Alfabet database. This includes the Alfabet Web Application, the Alfabet Server (service), Alfabet Expand or batch utilities.	to the database is required.
Create Configuration Meta-Model Update File	Save the customer configuration of the solution environment of the current database to an *.amm update file. For more information, see the section Administration Tasks Related to Solution Design.	Yes, via the **Connect** functionality in the context menu.
Update Assembly	Update the solution assembly from a DLL file (for example, for a new patch release). This is normally executed by Software AG Support. **NOTE**: This action modifies the content of the Alfabet database.	Yes, via the **Connect** functionality in the context menu.
Check SMTP	Test the connection to the SMTP server for outgoing mail from the Alfabet Server. Please note that this functionality is only available if.NET remoting is activated in the **Application Server** tab and no remote alias for connection to an application server is defined. For more information, see the section Testing the Connection to the SMTP Server.	Yes, via the **Connect** functionality in the context menu.
Register Event Logger	Registers the Alfabet component with the Windows® Event Log. After registration, login and logout actions are written to the Windows Event Log. Registration requires a specific configuration of the Alfabet component prior to performing the registration. For more information, see the section Tracking Login Actions in the Windows Event Log.	No

Functionalities Available via the Expanded Explorer of the Connected Alias

When you connect to a server alias using the **Connect** functionality in the context menu of the server alias, the explorer node of the server alias is expanded, allowing the following tasks to be performed. The tasks are described exclusively in the workflow-oriented part of this manual. A description of these complex functionalities is not repeated in this chapter.

Explorer Node	Functionality
Domain Accessibility Configuration	Allows you to configure authorization for Alfabet on a per domain or per user basis when using Windows sign-on. For more information, see the section Configuring Windows Sign-On. The settings are written to the AlfabetMS.xml configuration file.
External Sources Configuration	Allows to configure the interface for data synchronization with data from external sources. For more information, see Integrating Data from External Sources.
User Management	Allows you to configure new users and administrate user access to Alfabet functionalities. For more information, see the section Managing New and Existing Users. Changes are written to the instance store of the Alfabet database.
XML Objects	The configuration of the solution environment is in many cases done by configuring XML objects. These XML objects are normally configured with the configuration tool Alfabet Expand. If required, XML objects can be configured to be visible and editable in the Alfabet Administrator. If the **Visible In Administrator** attribute of an XML object is set to `True` in Alfabet Expand, you can see the XML object in the expanded explorer node of server alias configurations and edit it with the Alfabet Administrator. It is recommended that the configuration tool Alfabet Expand is used to configure XML objects because of the advanced editing functionalities available. For basic information about how to work with XML objects and for information about individual XML object configurations, see the reference manual *Configuring Alfabet with Alfabet Expand*. Changes to any XML object are written to the customer configuration in the Alfabet database. They affect only those objects in the database that are created after the configuration change. Typically, XML configurations should be edited before entering any data in the database.
Assemblies	Allows you to upload assemblies that are provided by Software AG to cover customer specific demands to the database. Changes are written to the solution store of the database.

Managing New and Existing Users

The **User Management** node of the Alfabet Administrator allows you to create new users and edit existing users in Alfabet. Once a user has been created, you can assign a user profile to it and administrate user passwords.

Typically, a user administrates the user password him/herself. However, if the user has forgotten the password, you can reset a password, clear the recent passwords list in order to allow their re-use, and change a password.

ⓘ The Alfabet Administrator only allows the administration of users accessing the Alfabet applica-
tion. Contacts (users with no access permissions) do not access the Alfabet application and are
not included in this view.

The functionalities in user management that are related to authentication and authorization are described
in the context of their function in the chapter Considering Security Issues.

You can do any of the following:

- Creating a New User

- Searching for an Existing User

- Deleting a User

- Assigning a User Profile to a New or Existing User

- Clearing a User's User Context Settings

Creating a New User

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer, right-click the relevant server
   alias and select **Connect**.

2) Expand the **User Management** node and click the **Users** node. The **User Management** view is
   displayed in the work area.

3) Click the **New** 🗈 button and select **Create User**. A window opens. Enter information in the fields,
   as needed:

   Basic Data tab:

   - **First Name**: Enter the user's first name. You may enter up to 128 characters.

   - **Name**: Enter the user's family name. This name is used for the definition of authorized users of
     objects, contact persons, owners of assignments, etc. You may enter up to 128 characters.

   - **Technical Name**: If necessary, enter a technical name for the user. If the Alfabet Web
     Application is configured to use the **Technical Name** attribute rather than the **Name** attribute
     for users, then your enterprise should define a technical name for all users.

     ⓘ The Alfabet Web Application can be configured to use the user's technical name in-
     stead of the user name in the **Audit History** functionality, which tracks the modifi-
     cation of Alfabet objects. If a technical name is not defined for a user, then the user
     name (**Name**, **First Name**) of that user is used in the **Audit History** functionality.
     For general information about the audit functionality, see the section *Viewing the
     Change History of an Object* in the reference manual *Getting Started with Alfabet*.
     For information about how to configure the auditing functionality, see the section
     Specifying History Tracking for an Object Class in the reference manual *Configuring
     Alfabet with Alfabet Expand*.

   - **Phone**: Enter a telephone number where this user can be reached. The user can change or
     update the phone number in the **Personal Info** functionality available in the < **Alfabet User
     Name** > menu.

- **Password Expiration Date**: Enter an expiration date for the password. The user's password will expire on this date and the user will need to define a new password to access Alfabet.

- **Change Password**: If you select this checkbox, the user must change his/her password during the first login. The user can subsequently change his/her password as needed by means of the **Change Password** functionality in the **<UserName>** menu. To define the user password, see the section *Defining, Clearing, and Resetting a User's Password*.

- **Type**: The user type specifies the available access permissions. Select `NamedUser`. Users of the type `NamedUser` must be assigned least one user profile (with either `ReadWrite` or `ReadOnly` access permissions). You can assign `NoAccess` to remove the access permissions from a user.

  > The user type `Anonymous` is used only for users that are automatically generated during login with single-sign on mechanisms. If the **Type** attribute is specified as `Anonymous` for a user, the system administrator should consider whether that user should be changed to a user of the type `NamedUser`. Anonymous users have `ReadOnly` access permissions via the user profile specified for anonymous users. For information about how to define an Alfabet user as a user of the type `NamedUser` or `Anonymous`, see the section Configuring User Authentication in the reference manual *System Administration*. For more information about specifying a user profile for anonymous users, see the section *Defining a User Profile for Anonymous Users*.

  > You can also create users of the stereotype **Contact** in the **Contact Administration** functionality. These users are created for documentation purposes only and will have no permission to access Alfabet. For example, an enterprise might want to manage roles associated with vendor personnel in order to manage vendor contacts, but would not want the vendor personal to have access to Alfabet. Ideally, new users that should have no access permissions should be created based on the person stereotype **Contact** in the **Contract Administration** functionality. For more information about creating a contact, see the section *Creating a Contact*.

- **User Name**: Enter a name that the user must enter when logging in to Alfabet. Because the user name required for login is not case-sensitive, the user name will be saved to the database in uppercase letters.

- **Picture**: Click the arrow to upload a picture to Alfabet. The picture can be in GIF, PNG, or JPG formats and may not be larger than 16 KB. The user can change or update the picture in the **Personal Info** functionality available in the < **User Name** > menu. The picture will be displayed in the context of the **My Collaborations** functionality and in the main toolbar of the Alfabet user interface next to the < **UserName** > menu. For more information about the collaboration capability, see the section *Communicating with Your Colleagues via the Alfabet Internal Collaboration Functionality* in the reference manual *Getting Started with Alfabet*.

- **General User Permissions**: Define the following user permissions for the user:

  - **Can Execute Batch Jobs**: Select the checkbox if the user may execute batch utilities designed to access the Alfabet database in stand-alone or remote mode. Access to batch utilities should only be allowed for administrators. For more information about batch utilities, see About Batch Utilities for Alfabet in the reference manual *System Administration*.

  - **Has Access to Diagram Designer**: Select the checkbox if the user may access the tool Alfabet Diagram Designer. If you select the checkbox, the **Open Diagram** button will be displayed in the diagram page views that the user has access permissions to. If you clear

the checkbox, the **Open Diagram** button will be displayed in the diagram page views that the user has access permissions to. For more information about to use the tool Alfabet Diagram Designer, see the reference manual *Designing IT Landscape Diagrams in Alfabet*.

- **Exclude from Anonymization**: Select the checkbox if the user may not be anonymized by means of the data anonymization capability.

- **Is User Assistant**: Select the checkbox if the user is responsible for answering user questions about the Alfabet solution. Users can request assistance about a standard or configured view in Alfabet via a **Request User Assistance** option in the Help menu available in the main toolbar in Alfabet. If triggered, an email with a link to the view will be sent to the user in the enterprise who has been specified as responsible for user assistance

> . The **Request User Assistance** option must be enabled in the XML object *SolutionOptions*. For more information, see the section *Enabling the User Assistance Functionality* in the reference manual *Configuring Alfabet with Alfabet Expand*. For a description of the **Request User Assistance** option, see the section *Requesting User Assistance for a View in the User Interface* in the reference manual *Getting Started with Alfabet*.

Mandates tab: Select a checkbox to assign one or more mandates to the selected user. Typically, a user will be assigned only one mandate. The user will be able to view only objects that have been assigned to his/her mandate. The first mandate checked in the **Mandates** tab of the **User** editor is, by default, the mandate automatically used at login. For an overview of the access permission concepts implemented in Alfabet, see the section *Understanding Access Permissions in Alfabet* in the reference manual *Getting Started with Alfabet*. A user can change mandates at any time during their Alfabet session. For more information about how a user can change the mandate he/she is currently working with, see the section *Changing the Mandate That You Are Logged In With* in the reference manual *Getting Started with Alfabet*. For more information about the configuration of mandates, see *Implementing the Mandate Capability for a Federated Architecture* in the reference manual *Configuring Alfabet with Alfabet Expand*.

Select the checkbox for the **Mandate Master** attribute to specify the selected user as a mandate master. A user that has the **Mandate Master** attribute checked can view any object regardless of its mandate.

API Permissions tab:

- **Has API V2 Access**: Select the checkbox if the user may send requests to the Alfabet RESTful API V2. These settings are only required for a user that is technically used by a RESTful client for authorization against the RESTful interface of Alfabet Web Application, version 2. The **Has API V2 Access** checkbox should be deselected for security reasons for all other users. For more information about the Alfabet RESTful API and the required settings in this tab to configure access to the Alfabet RESTful API, see the reference manual *Alfabet RESTful API*.

Alfabet Expand Permissions tab:

- **Has Access to Alfabet Expand**: Select the checkbox if the user may access the configuration tool Alfabet Expand. Access to Alfabet Expand should only be allowed for solution designers and administrators. The **Type** field in the **Basic Data** tab must be set to `NamedUser` in order to allow access to Alfabet Expand. You should specify the individual functionalities that the user can work with in the **Expand Access Options** field. For more information about the configuration capabilities available in Alfabet Expand, see the reference manual *Configuring Alfabet with Alfabet Expand*.

- **Alfabet Expand Access Options**: Select each checkbox to specify which functionalities the user should be allowed to access in the configuration tool Alfabet Expand. Only users of the type `NamedUser` may access Alfabet Expand, execute batch jobs, and access the Alfabet Diagram Designer. Please note the following:

Designation in Alfabet Expand Web	Designation in Alfabet Expand Windows
ADIF Designer	**ADIF** tab
Administrator	**Admin** tab
Business Function Designer	**Functions** tab
Class Designer	**Meta-Model** tab
Diagram Model/Shape Designer	not relevant
Event Designer	**Events** tab
Guide Page Designer	**Guide Page Designer** option in **Managers Menu**
Icon Designer	**Icon** node in **Presentation** tab
Not relevant	**Managers Menu** containing **Help Manager**, **Database Manager**, etc.
Presentation Model Designer	**Presentation** tab
Publication Designer	**Publications** tab
Report Designer	**Reports** tab
Not relevant	**Survey** tab
System Administrator	not relevant

Designation in Alfabet Expand Web	Designation in Alfabet Expand Windows
Utilities	**Meta-Model** menu and **Globalization** menu
Workflow Designer	**Workflows** tab

Collaboration Data tab:

- **Email**: Enter the user's email address. The email address is necessary for notification emails sent in the context of various Alfabet functionalities including, for example, assignments and monitors. The user can change or update the email address in the **Personal Info** functionality available in the < **UserName** > menu.

- **Email Notification Language**: Select the language to be used for the text messages in automatically-generated email notifications. The default language is English.

  > For more information about the configuration of language versions for automatically-generated email notifications, see the section *Configuring Text Templates for Email Notifications* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- **Skype ID**: Enter the user's Skype ID to implement interoperability with Skype. A maximum of 128 characters is allowed. If interoperability with Skype is enabled for the user, a Skype presence status icon will be displayed next to the user's name in the **Attributes** section of object profiles/object cockpits and previews, allowing other users to contact the authorized user of an object should questions arise. Integration with Skype for Business Server® must be configured in order to implement Skype in the context of Alfabet.

  > The **Skype ID** and **Skype Domain** attributes will only be displayed in the **User** editor and in **Users Administration** functionality if interoperability with Skype for Business Server® is activated. For more information about using the Skype capability to communicate with your colleagues, see the section *Skyping with Your Colleagues*. For more information about configuring integration with Skype, see the section *Configuring Interoperability with Skype for Business Server®* in the reference manual *API Integration with Third-Party Components*.

- **Skype Domain**: Enter the user's Skype domain to implement interoperability with Skype.

- **MS Teams User Name**: If integration with Microsoft® Teams is supported in your solution, enter the user's user name in MS Teams.

- **MS Teams User ID**: Enter the user's user ID in MS Teams.

4) Click **OK** to save the data or click **Cancel** to exit without saving your changes. The new user is now visible in the table.

Searching for an Existing User

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer, right-click the relevant server alias and select **Connect**.

2) Expand the **User Management** node and click the **Users** node. The **User Management** view is displayed in the work area.

3) In the **User Management** page view, enter search parameters in the **Search Pattern** field. Additionally, you can define attributes and/or a user profile as search criteria.

4) Click **Update**. The search results are displayed in the table.

Deleting a User

You can delete a user from the Alfabet database.

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer, right-click the relevant server alias and select **Connect**.

2) Expand the **User Management** node and click the **Users** node. The **User Management** view is displayed in the work area.

3) In the table, click the user you want to delete.

4) In the toolbar, click the **Delete** 🗑 button.

5) Confirm the warning by clicking **Yes,** or click **No** to exit without deleting the selected object(s).

Assigning a User Profile to a New or Existing User

Once a user is created, you can assign a user profile to it. A user profile defines the scope of functionality available to the user. User profiles are created and configured in the configuration tool Alfabet Expand.

> ⓘ You can only assign user profiles to users of the type `NamedUser`. If you assign profiles to an anonymous user, your settings are ignored, and the user is assigned the default profiles for anonymous access.

1) In the table, click the user that you want to attach profiles to.

2) In the toolbar, click **User Profiles...**.

3) In the window that opens, select the checkbox of all profiles that you want to assign to the user and click **OK**.

Clearing a User's User Context Settings

User context settings are saved in the database. A user's user context settings include, for example, filters defined in Alfabet views or search parameters used for the most recent search. When a user returns to a Alfabet view or the search functionality in following user sessions, such filter definitions or search parameters have been saved and are automatically displayed in the respective field.

User context settings can be cleared for individual users in the **User Management** functionality in the Alfabet Administrator. After the user context settings have been cleared, relevant filters in views or fields in the search functionalities will be empty and new user context settings can be saved.

To clear a user's user context settings:

1) In the table, click the user for whom you want to clear the context settings.

> ⓘ You can select several objects in the table at the same time by holding down the CTRL key while selecting.

2) In the toolbar, click **Action** > **Clear User Context Settings**.

3) Confirm the warning. The user context settings of the selected user are cleared.

Usage Tracking

When your license for using Alfabet is based on a metered contract, it is required that the usage of functionalities is tracked and data about the usage is provided to Software AG.

This tracking process requires a special configuration during or after the installation of the Alfabet Server. This section describes the tracking process and the configuration and visualization.

About Usage Tracking

When usage tracking is activated, the Alfabet Server stores information about the usage of Alfabet functionalities. This information is written to a.dat file named `TrackUsage_<YYYY_MM>`.dat between 23:00 and midnight each day and at startup and shutdown of the Alfabet Server. A new file is only created the first day of each month. On the following days of the month, the Alfabet Server adds entries to the existing file. To fulfill a metered contract, you must send the closed file to Software AG each time a new file has been started.

The.dat files are stored encrypted in a configurable directory. By default, this is the working directory of the Alfabet Server. The Alfabet Server (Service) must have write access permissions to the target directory to store tracking data.

The Alfabet Server writes a tracking record whenever a user logs in with a specific user profile or changes the user profile. To ensure the privacy of the user, the `REFSTR` of the user is written to the file instead of the name or user name. The tracking records are evaluated within the contractual tracking period in the following way:

- If a user is logged in with ReadOnly user profiles only during the tracking period, he/she is counted as one *Analysis User*. This is independent of the overall number of user profiles that the user logged in with during the tracking period and independent of the overall number of logins for the user during the tracking period.

- If the user is logged in with both ReadOnly and ReadWrite user profiles or with ReadWrite user profiles only, he/she is counted as one *Business User*. This is independent of the overall number of user profiles that the user logged in with during the tracking period and independent of the overall number of logins for the user during the tracking period.

- If a user logged in 25 days or less during the tracking period, he/she is additionally marked as an occasional user either for the *Business User* or *Analysis User* count, depending on the access permissions of the user profiles he/she logged in with during the tracking period.

The license model for usage tracking was different for contracts purchased with Alfabet 9.6 or with previous Alfabet versions:

The Alfabet Server wrote a tracking record for the first time each day that a specific user accessed a specific functionality or configured report. There were three types of usage types in Alfabet: *Viewer* for access with ReadOnly access permissions, *Data Entry User* for ReadWrite access to simple data maintenance functionalities, *Functional User* for ReadWrite access to sophisticated Alfabet functionalities. If the same user accessed another functionality or configured report of the same usage type on the same day, no records were written. Usage tracking was independent of the user profile that the user was logged in with. If the user accessed a functionality of the same usage type on the same day with a different user profile, no new record was written to the file. Tracked information is limited to the contractual information regarding how many users used at least one functionality with a given usage type per day.

The information in the .dat files is stored encrypted. Alfabet provides an interface to allow customers to view the tracking information written to the file. Access to the information stored in the files is only possible via this interface.

Files can be read by the Alfabet Administrator and the result of usage tracking is then displayed on the interface of the Alfabet Administrator. The access permissions and the number of users assigned to the user profile are also displayed.

Activating Usage Tracking

To perform usage tracking, a running Alfabet Server must be connected to the same Alfabet database than the Alfabet Web Application. The Alfabet Server can be run as a service.

Usage tracking must be activated in both the server alias of the Alfabet Web Application and the server alias of the Alfabet Server.

Do the following to activate usage tracking in the server alias configuration of the Alfabet Web Application and the Alfabet Server:

1) Define a target directory for the storage of tracking data and define write access permissions for the Alfabet Server (Service) for the directory.

2) Open the Alfabet Administrator and click the **Alfabet Aliases** node.

3) In the table, select the server alias of the Alfabet Web Application that you want to track usage for and click the **Edit** button in the toolbar. You will see the editor in which you can edit the server alias.

4) Go to the **Server Settings** > **Tracking** tab.

5) Select the **Track Usage** checkbox.

6) Click **OK** to save your changes.

7) In the table, select the server alias of the Alfabet Server that you want to use for usage tracking and click the **Edit** button.

8) Go to the **Server Settings** > **Tracking** tab.

9) Select the **Track Usage** checkbox.

10) In the **Track Usage Directory** field, either select the target directory using the **Browse** ![button] button, or specify a path to the storage location absolute or relative to the working directory of the Alfabet Server.

> ⓘ Server variables can be used to define part of the information as a variable with the value set in the **Variables** tab of the server alias configuration. For more information, see [Defining Connections Based On Server Variables](#).

11) Click **OK** to save your changes.

Viewing the Tracking Information

You can view information about the contract-relevant usage of Alfabet based on the information written to the.dat file in the interface of the Alfabet Administrator. In the report, the number of users that accessed Alfabet functionalities with a distinct contract-relevant usage type is displayed as well as the number of users that logged in to the Mobile Portfolio Manager.

The Alfabet Administrator reads the.dat files in the usage tracking directory. Therefore, the following usage tracking information is excluded from display:

- information for the current day that is not yet written to a.dat file, and

- usage tracking records of.dat files deleted or archived in another directory than the usage tracking directory.

- The **Occasional Usage Count (last 12 month max)** column displays the subset of users counted in the **Usage Count** column that are occasional users (users only logged on 25 days or less for the respective combination of **Usage Type**, user name and day for the tracking period defined with the **From Date** and **To Date** filter fields). If the tracking period exceeds 12 months, occasional users will only be evaluated for the last twelve months of the tracking period.

To view the tracking information:

1) Click the **Usage Tracking** node in the **Administrator** explorer. A blank page with a filter on top is displayed on the right.

2) Define the following filters:

- **Select Alias**: Select the server alias of the Alfabet Server that is used for usage tracking.

- **Select License Model**: Select the license model that applies to your contract. Contracts that were purchased with Alfabet 9.6 or previous versions of Alfabet count usage based on business function and configured report usage, while contracts for Alfabet 9.7 and higher count usage based on the access permissions for user profiles that a user logged in with.

- **From Date**: Select the start date of the period that you want to view the usage tracking information for.

- **To Date**: Select the end date of the period that you want to view the usage tracking information for.

3) Click **Update** to view the report.

The report shows the following data for the new license model purchased with Alfabet 9.7 or higher:

- **Usage Type**: License access permission definition for the user profiles that the users are logged in with. There is a line for each of the usage types `Analyis User` (logged in with ReadOnly user profiles only), and `Business User` (logged in with at least one user profile with ReadWrite access permissions).

- **Usage Count**: The usage count is displayed as the maximum number of distinct users logged in with either ReadOnly or ReadWrite user profiles in the period defined by the filter setting for the report. For example, if you select a period of three days, and on the first day users A and B and on the other two days of the period users B and C were logged in with a ReadOnly user profile only, the **Usage Count** column for `Analysis User` displays 3 to inform you that three different users were logged in with ReadOnly user profiles only in the defined period.

  Please note that the report counts each user only once for the highest usage type in the hierarchy of usage types (`Analysis User` < `Business User`) accessed by the user. That means that a user logged in with both a ReadOnly and a ReadWrite user profile in the same period is only counted in the column for the usage type `Business User`. The access to the functionality of the usage type `Analysis User` is ignored in the tracking report.

- **Occasional Usage Count (last 12 months max)**: The subset of users listed in the **Usage Count** column that have not logged in on 25 days or less during the defined period. If a tracking period of more than 12 months is selected for the report, the occasional usage count is only calculated for the last twelve months within the selected period.

The report shows the following data for the old license model purchased with Alfabet 9.6 or previous versions of Alfabet:

- **Usage Type**: Displays the license access permission definition for the accessed functionality. There is a line for each of the usage types of `Viewer`, `Data Entry User` and `Functional User`.

- **Usage Count**: Displays the usage count is displayed as maximum number of distinct users working on the level of a given usage type in the period defined by the filter setting for the report. For example, if you select a period of three days and on the first day users A and B and the on the other two days of the period users B and C accessed a functionality of the usage type `Data Entry User`, the **Usage Count** column would display 3 to inform you that three different users were using a `Data Entry User` functionality in the defined period.

  Please note that the report counts each user only once for the highest usage type in the hierarchy of usage types (`Viewer` < `Data Entry User` < `Functional User`) accessed by the user. That means that a user that accessed both a functionality of the usage type `Data Entry` and `Functional User` in the same period is only counted in the column for the usage type `Functional User`. The access to the functionality of the usage type `Data Entry User` is ignored in the tracking report.

- **Occasional Usage Count (last 12 months max)**: Displays the subset of users listed in the **Usage Count** column that have not logged in on 25 days or less during the defined period. If a tracking period of more than 12 months is selected for the report, the occasional usage count is only calculated for the most recent twelfth month within the selected period.

If your specified time period for the report includes a month for which no usage tracking file is available, the information will be highlighted red in the report:

FIGURE: Usage tracking report for the new license model with a warning that a file is not found

Presentation Usage Tracking

The user activity in the Alfabet user interface can be tracked to evaluate which functionalities are most used or seldom used and how users navigate on the Alfabet user interface to access a functionality. In addition, the information about the performance of data preparation for view rendering is captured. The information from activity tracking can help to improve the solution configuration and to identify performance issues.

The following information is available:

- [Activating Presentation Usage Tracking](#)

- [Reading the Presentation Usage Tracking Information](#)

Activating Presentation Usage Tracking

Presentation usage tracking needs to be activated in the server alias of the Alfabet Web Application:

1) Expand the **Alfabet Aliases** node in the **Administrator** explorer. The right pane displays a list of all available alias configurations.

2) In the table, select the alias configuration that you want to edit and click the **Edit** ✎ button. You will see the editor in which you can edit the alias configuration.

3) Go to the **Server Settings** > **Tracking** tab and edit the following attributes:

   - **Track Presentation Usage**: Select **Local Database** to activate presentation usage tracking. By default, this attribute is set to **Deactivated** and no presentation usage tracking is performed. The **Alfabet REST Service** mode is currently not supported.

   - **Track User ID**: If presentation usage tracking is activated and this checkbox is selected, the `REFSTR` of the user will be tracked in addition to the session ID of the current user session and sub-session. If the checkbox is not selected, only the session ID and sub-session ID of the current user session will be saved in the tracking records. This information is sufficient to evaluate which views have been accessed by a user during the same session without adding actual user information to the tracking information. Check legal compliance of tracking user information prior to selecting the **Track User ID** checkbox.

   - **Archive/Restore Presentation Usage Tracking**: If you select the checkbox, the presentation tracking information will be included in Alfabet Database Archive files (ADBZ files) and restored in target databases when restore is performed from an ADBZ file. The presentation

usage tracking feature and this attribute must be activated on both the alias used for archiving the database and the alias for restoring the database to include the information.

4) Click **OK** to save your changes.

Reading the Presentation Usage Tracking Information

If presentation usage tracking is activated in the server alias of the Alfabet Web Application, information about all user activity will be written to the `ALFA_PRES_USAGE_TRACKING` class table of the Alfabet database. All time information is returned in Web server time.

This table is not visible in **Meta-Model** tab of Alfabet Expand. Nevertheless, configured reports based on native SQL can be defined to read the data.

The `ALFA_PRES_USAGE_TRACKING` table has the following columns:

- `TRACK_ID`: A unique ID for this tracking information.

- `PRESENTATION_TYPE`: The type of view the user is working with. For example, `ObjectCockpit` for an object cockpit, `BusinessFunction` for a functionality, or `Wizard` for a wizard.

- `PRESENTATION_NAME`: The name of the view the user is working with.

- `ENTRYPOINT_TYPE`: The entry point can have the following values:

  - `MainMenu` for access via a guide page, guide view or a menu defined for a user profile.

  - `Bookmark` for access via a bookmark.

  - `ExternalLink` for external access (for example, via an express view email or emails sent out for assignments).

- `ENTRYPOINT_NAME`: For access via bookmarks, the name of the bookmark is stored. For access via an external link, the accessed view is stored. If the `ENTRYPOINT_TYPE` is `MainMenu`, this column is empty.

- `NAVIGATION_SOURCE_TRACKID`: The navigation source track ID is identical for all objects that have been opened simultanously. If a user accesses an object cockpit, all configured reports that are embedded in the object cockpit and are therefore visible to the user will be tracked. The object cockpit and the configured reports will have the same navigation source track ID. If a user is accessing a view multiple times, the navigation track ID is different.

- `NAVIGATION_SOURCE_TYPE`: The type of view the user was opening this view from. For example, `ObjectCockpit` for an object cockpit or `BusinessFunction` for a functionality.

- `NAVIGATION_SOURCE_NAME`: The name of the view the user was opening this view from.

- `BASEOBJECT_ID`: The REFSTR of the base object the user is currently working with.

- `BASEOBJECT_NAME`: The name of the base object the user is currently working with.

- `BASEOBJECT_CLASS`: The name of object class of the base object the user is currently working with.

- `SESSIONS_ID`: The session ID of the user session. If the user logs out and logs in again, the session ID will change. The session ID does not inform about the user that logged in.

- SUBSESSION_ID: The sub-session ID of the user session. A new sub-session ID is provided if a view is opened in a new tab such as when opening a diagram in the Alfabet Diagram Designer.

- USER_ID: If the **Track User ID** checkbox in the server settings of the Alfabet Web Application is set, the REFSTR of the user accessing the view will be tracked. Otherwise this column is empty.

- USER_PROFILE: The name of the user profile the user is logged in with.

- START_TIME: The time when the user accesses the view.

- END_TIME: The time when the user leaves the view.

- CONTEXT_INFO: This column is currently not filled.

- DETAILS: This column is currently not filled.

- POLICY_ID: This column is currently not filled.

- PREPARE_TIME: This column stores the time when data preparation is finished and rendering of the view is started.

- PREPARE_DURATION: This column returns the time period between the time the user requested the view and the time data preparation has finished. This information can be used to identify performance problems with for example configured reports that are due to query execution. Customers can build configured reports vizualising view preparation performance according to their requirements.

Index