Alfabet Expand Web Specific Administration Tasks

Alfabet Reference Manual

Documentation Version Alfabet 10.15.2

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.)
	This is a note providing additional information.
	This is a note providing procedural information.
	This is a note providing an example.
	This is a note providing warning information.

Table of Contents

Chapter 1: Introduction

Alfabet Expand Web is designed for Cloud services and installed exclusively by Software AG. Therefore, information required to implement features and perform administrative tasks is not included in the reference manual *System Administration* that is written for customers with on-premise installations.

You can access the following functionalities by clicking a tile in the start page or selecting the relevant functionality in the **Expand Designers** menu:

1) **ADIF Designer**: Configure ADIF schemes for the batch import, export, and manipulation of large amounts of data in the Alfabet database using the Alfabet Data Integration Framework (ADIF).

2) **Administrator**: Configure user profiles, mandates, business functions, and custom explorers for the user community.

3) **Class Designer**: Configure object class properties, object classes, enumerations, cultures, API cultures, and database views.

4) **Diagram Model Designer**: Configure custom diagram definitions and custom diagram item templates to use when designing diagrams in the Alfabet Diagram Designer

5) **Diagram Shape Designer**: Design the visualization of the custom diagram item template that is currently selected in the **Diagram Model Designer**.

6) **Events**: Configure event templates in order to specify events that trigger ADIF import/export schemes, workflows, REST API calls, and other relevant Alfabet functionalities.

7) **Guide Page Designer**: Configure navigation pages as start pages for your user community.

8) **Icon Gallery**: Upload custom icons to Alfabet and create icon groups bundling icons to use in various Alfabet functionalities.

9) **Presentation Model Designer**: Configure editors, wizards, selectors, object views, workflows, class settings, GUI schemes, text templates, monitor templates, conditions, and various XML objects providing functionalities in Alfabet.

10) **Publication Manager**: Configure the export of data to Microsoft® Word-based publications.

11) **Report Designer**: Configure reports relevant for your user community as well as various Alfabet functionalities.

12) **System Administrator**: Specify the identity provider certificate file for federated single sign-on configuration.

13) **Utilities**: Save the configuration of the Alfabet solution to an AMM file. Manage automated assistants and vocabularies.

14) **View Designer**: Design the visualization of the editor, object cockpit, or graphic view that is currently selected in the **Presentation Model Designer**.

15) **Workflow Designer**: Configure workflow templates in order to make Alfabet 's workflow functionality available to your user community.

16) **Workflow Diagram Designer**: Design the visualization of the workflow diagram that is currently selected in the **Workflow Designer**.

For more information about access permissions to the various functionalities available in Alfabet Expand Web, see the section Accessing Alfabet Expand Web.

This short guide should provides an overview of the additional configuration and installation requirements for Alfabet Expand Web for internal use.

The following information is available:

- [Configuring Access to the Web version of Alfabet Expand and the Guide Pages Designer](#)

- [Configuring the Alfabet Web Application to Take Over Configuration Data From A Master Database](#)

- [Configuring the Alfabet Web Application to Act as a SAML Service Provider](#)

Chapter 2:   Installation

Alfabet Expand Web is an integrated functionality of the Alfabet Web Application and can be accessed via opening a URL in a web browser. The technical requirements are identical to the technical requirements for the Alfabet Web Application.
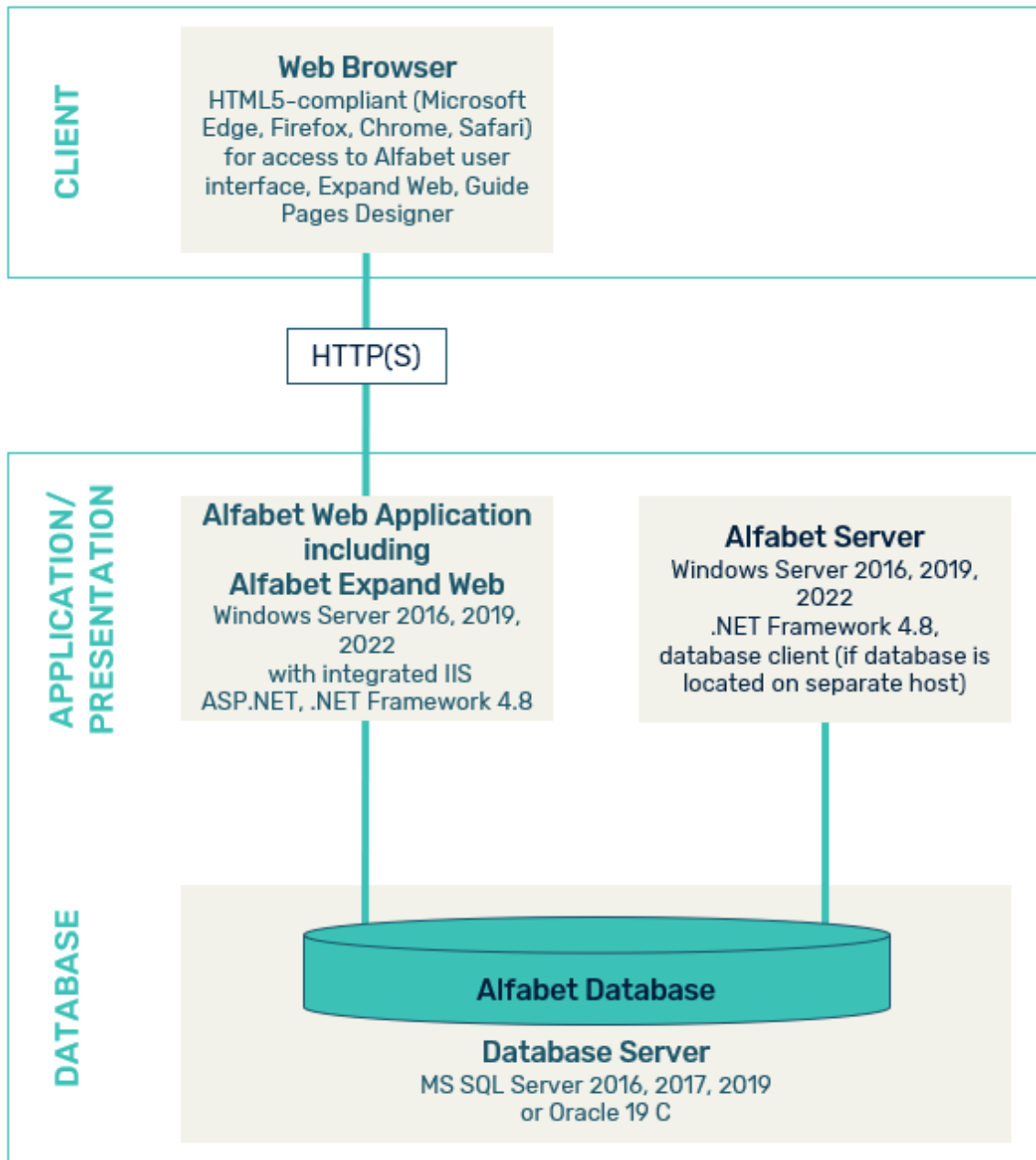
This section provides the information to implement a working version of Alfabet Expand Web on an already installed Alfabet Web Application. The basic installation of the Alfabet Web Application is described in the reference manual *System Administration* in the chapter *Installation* and not repeated here.

Depending on the functionalities that you would like to use, you must follow different installation steps in addition to the basic installation of the Alfabet components required for working with the Alfabet Web Application:

• In any case, you must follow the instruction Configuring Access to the Web version of Alfabet Expand and the Guide Pages Designer. This will activate most of the functionalities except data anonymization and update of the meta-model configuration from a master database.

- You must additionally configure the Alfabet Web Application to connect to a master Alfabet database with a separate server alias if you would like to take over all or part of the meta-model configuration from a master database.
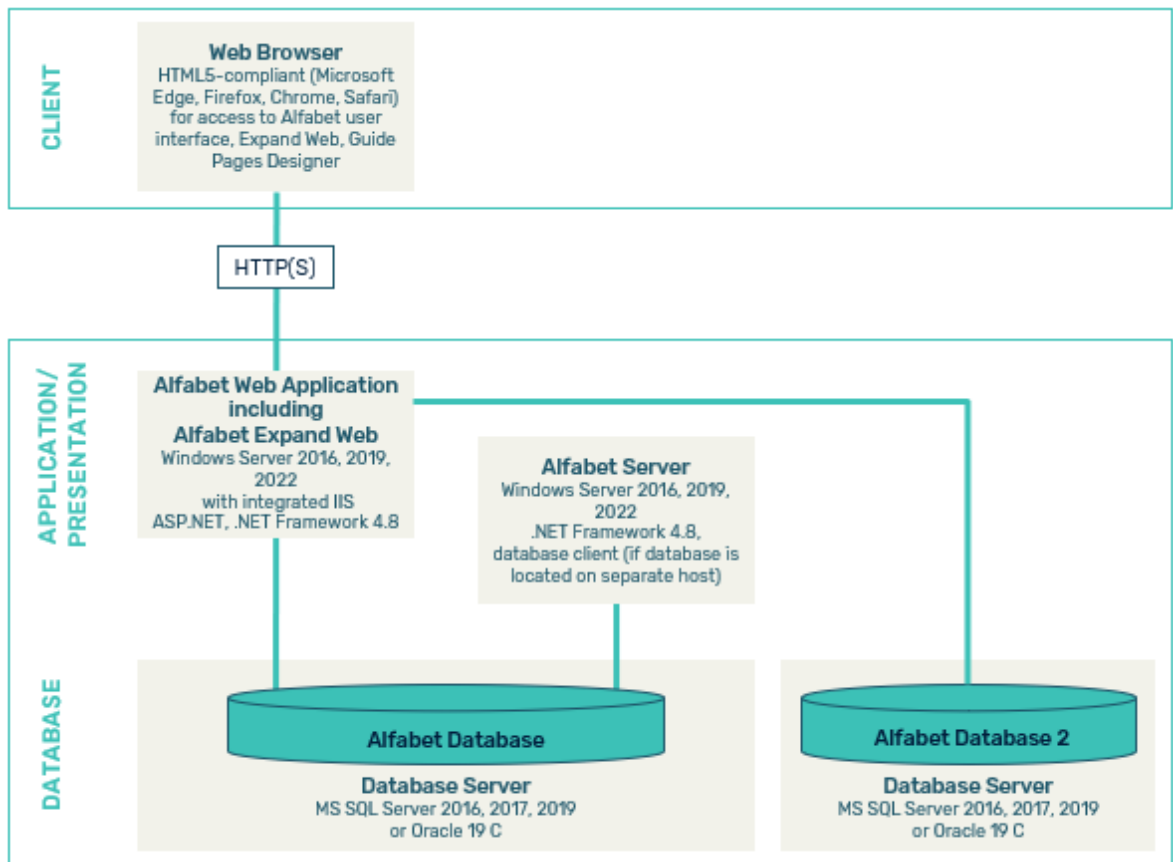
Configuring Access to the Web version of Alfabet Expand and the Guide Pages Designer

The Web version of Alfabet Expand and the Guide Pages Designer are part of the Alfabet Web Application and use the same `web.config` files and server alias configuration.

A special license must be added to the server alias of the Alfabet Web Application to access the Web version of Alfabet Expand and the Guide Pages Designer. If your company has licensed these components, the license key will be provided by Software AG.

If you have only licensed the Guide Pages Designer, the license key will be automatically added to the server alias configuration the first time you open the server alias editor in the Alfabet Administrator.

If your license also contains access to Alfabet Expand Web, the license must be included in the server alias using the Alfabet Administrator:

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

2) In the table on the right, select the server alias of the Alfabet Web Application and click the

   Edit 🖉 button in the toolbar. An editor opens.

3) In the editor, go to the **License** tab.

4) Enter the license key in the field in the **Key** sub-tab.

5) Go to the **Summary** sub-tab. In the field, a description about the scope of the license shall be displayed. If an error message is displayed, the license key is invalid or a typo has been added to the key while entering it into the **Key** sub-tab.

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

Configuring the Alfabet Web Application to Take Over Configuration Data From A Master Database

To configure and apply new meta-model configurations to the Alfabet application, configuration should typically not be carried out in the production environment but rather in 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 from the configuration to the test environment, for example, Alfabet Expand offers a mechanism that reads data directly from the configuration database (which shall be referred to as the "master database") to the current database of the test environment in the following called target database (which shall be referred to as the "target database").

This mechanism requires that the following components are running:

- The Alfabet Web Application including a running Alfabet Expand Web connecting to the target database.

- An Alfabet Server connecting to the target database.

If the two environments are located on different hosts, the alias configurations centrally defined in the `AlfabetMS.xml` accessed via the Alfabet Administrator can then be written to individual `AlfabetMS.xml` files for each application. Please note that in this context the following alias configuration must be available in the `AlfabetMS.xml` used by the Alfabet Web Application of the target database:

- The server alias of the Alfabet Web Application including an Alfabet Expand Web license for connection to the target database.

- A remote alias for connection to the Alfabet Server.

- A server alias for connection to the master database. This server alias does not require an Alfabet Expand Web license.

The connection to the master database must be defined in the server alias of the Alfabet Web Application. Configuration is done in the Alfabet Administrator:

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

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) Go to the **Expand** tab.

4) Select the server alias for connection 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 performed with Alfabet Expand Web. Only selected reference data that has been configured in the **Configuration** functionalities in the Alfabet user interface and that are relevant for the configuration can be selected manually to be included in the configuration update. 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. This option should be selected if, for example, 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 a single 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 overwrite the target configuration. This option should be selected if, for example, 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.

7) Click **OK** to save your changes and close the Alfabet Administrator.

8) Access Alfabet Expand Web and select **META-MODEL** > **Restart Web Application**. The changes in the server alias configuration are only applied after restart of the Alfabet Web Application.

Configuring the Alfabet Web Application to Act as a SAML Service Provider

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

To ease the implementation of authentication via a SAML system for a Cloud solution, a web interface has been implemented in Alfabet Expand web. The functionality is only available if the license for Alfabet Expand Web includes the **System Administrator** designer.

The web interface only manages the part of the required configuration concerning the identify provider data. Additional configuration is required that cannot be performed by Cloud customers and must be carried out by Software AG Support prior to and after the configuration done by the customer in Alfabet Expand Web.

The following configuration steps have to be performed by Software AG Support prior to the SAML configuration via Alfabet Expand Web:

1) Copy the file `web.sso.saml2.config` from the subdirectory **Example** of the Alfabet Web Application directly into the physical directory of the Alfabet Web Application and change the name of the file to `web.config`.

> ⓘ 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 avoid overwriting it and then take over all required settings for other features from the old into the new `web.config` file via a text editor.
>
> The `alfabet.config` and `AppSettings.config` files must already be available in the subdirectory **config** of the physical directory of the Alfabet Web Application.

2) Open the Alfabet Administrator.

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

4) In the table, click the server alias of the Alfabet Web Application. An editor opens.

5) Go to the **Client Settings** > **Authentication** tab.

6) Set the attribute **Mode** to `SSO_FederatedAuthentication`.

7) Click **OK**. The change is saved and the editor is closed.

The following configuration can then be done by the customer. Please note however that it is recommended to send the identity provider certificate file and password to Software AG Support to do the complete configuration on server side.

Make sure that the identity provider certificate file is located on the local file system in a folder accessible via Alfabet Expand Web. The identity provider meta-data can either be provided via an URL, that must be accessible via Alfabet Expand, or via a file located on the local file system in a folder accessible via Alfabet Expand Web.

> ⓘ Note the following:
>
> - The service provider certificate and certificate password must be XML-conform and must not include any XML reserved characters like < or >.
>
> - Certificate signature must be based on SHA-1, SHA-256, SHA-384 and SHA-512.

1) In Alfabet Expand Web, open the **System Administrator** designer.

2) Click on **Federated SSO Configuration** in the explorer. The attribute window for **Federated SSO Configuration** opens.

3) Do either one of the following:

- In the **Identity Provider Metadata URL** attribute, enter the URL of the IDP Entity Provider Metadata Export.

- Click in the empty field of the **Identity Provider Metadata File** attribute. In the window that opens, click **Select File**, select the service provider metadata file, and click **Upload** to import the file.

4) Click in the empty field of the attribute **Service Provider Certificate File**.

5) In the window that opens, click **Select File**, select the service provider certificate file and click **Upload** to import the certificate.

6) In the **Service Provider Certificate Password** attribute, enter the password for the certificate.

7) In the explorer, click on the arrow on the right of **Federated SSO Configuration** node and select **Generate Configuration** from the context menu. The required files for SAML are created in a subfolder of the working directory of the Alfabet Web Application.

8) Part of the configuration cannot be changed via the web interface. After having used the **Generate Configuration** option, a message will be displayed with information about the required additional configuration steps. Send the message to Software AG Support.

The final configuration has to be done by a person with access to the physical directory of the installation, following the instructions provided via the interface.

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.

Chapter 3:   Accessing Alfabet Expand Web

After having configured the server alias to allow access to Alfabet Expand Web as described in the section Instal-lation, named Alfabet users with the required access permissions can access Alfabet Expand Web and the Web version of the Guide Pages Designer.

To grant a user access permission for Alfabet Expand: activate the check box **Has Access to Alfabet Ex-pand** in the **User** editor for 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*.

1) In the Alfabet user interface, go to the **Users Administration** functionality.

2) Select the user in the table and click **Edit** in the toolbar.

3) In the **User** editor, go to the **Alfabet Expand Permissions** tab.

4) Select the **Has Access to Alfabet Expand** checkbox.

5) In the **Alfabet Expand Access Options** field, select all Alfabet Expand designers that the user shall be able to access.

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

Users can access Alfabet Expand Web and the Guide Pages Designer via the following URL:

```
http://<URL of the Alfabet Web Application>/Expand.aspx
```

Chapter 4:   Executing Administrative Tasks in Alfabet Expand

This chapter describes the tasks that are not directly related to the configuration of an Alfabet functionality but are required to administrate the configurations performed with Alfabet Expand and to support system administration tasks.

The following administrative tasks can be carried out in Alfabet Expand:

- Applying Configuration Changes to Other Databases

- Anonymizing Data

- Configuring Default User Settings for the User Community

- Configuring the Use of External Sources with Alfabet

- Configuring the Visibility of Tabs in Alfabet Expand

Applying Configuration Changes to Other Databases

The Alfabet database is the basis of the Alfabet software. Software AG provides mechanisms to archive part or all of the configuration performed in an Alfabet database and restore it to another Alfabet database. These mechanisms enable configuration based on a copy of a production database in a configuration environment as well as the testing of the new configuration prior to implementing it in the production environment.

Software AG provides the proprietary file format AMM for storing the configuration of an Alfabet solution in one Alfabet database and restoring it in another Alfabet database. This is the central mechanism for configuration exchange. For special data types that cannot be included in AMM files, the **Import Data Search** capability can be used to import data from one Alfabet database to another Alfabet database.

The following information is available:

- About the Storage of the Configuration in the Alfabet database

- About Storage of Changes to the Meta-Model Performed with Alfabet Expand in the Alfabet database

- Overview of Administrative Tasks Related to Solution Design

- Taking Over Configurations Performed in Alfabet Expand With AMM Based Mechanisms

  - Identifying Configuration Objects via Solution Tagging

    - Setting Tags for a Single Configuration Object

    - Setting or Removing Tags For Multiple Configuration Objects Simultaneously

    - Setting a Default Tag Automatically Applied to New and Changed Objects

  - Versioning of Configurations

  - Saving the Configuration of the Alfabet Solution to an AMM File

    - Saving Complete or Tagged Types of Configuration Objects or the Complete Configuration with Alfabet Expand

    - Saving Selected Objects of the Configuration with Alfabet Expand

About the Storage of the Configuration in the Alfabet database

The Alfabet 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 *Saving the Configuration of the Alfabet Solution to an AMM File*.

- 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 to 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.
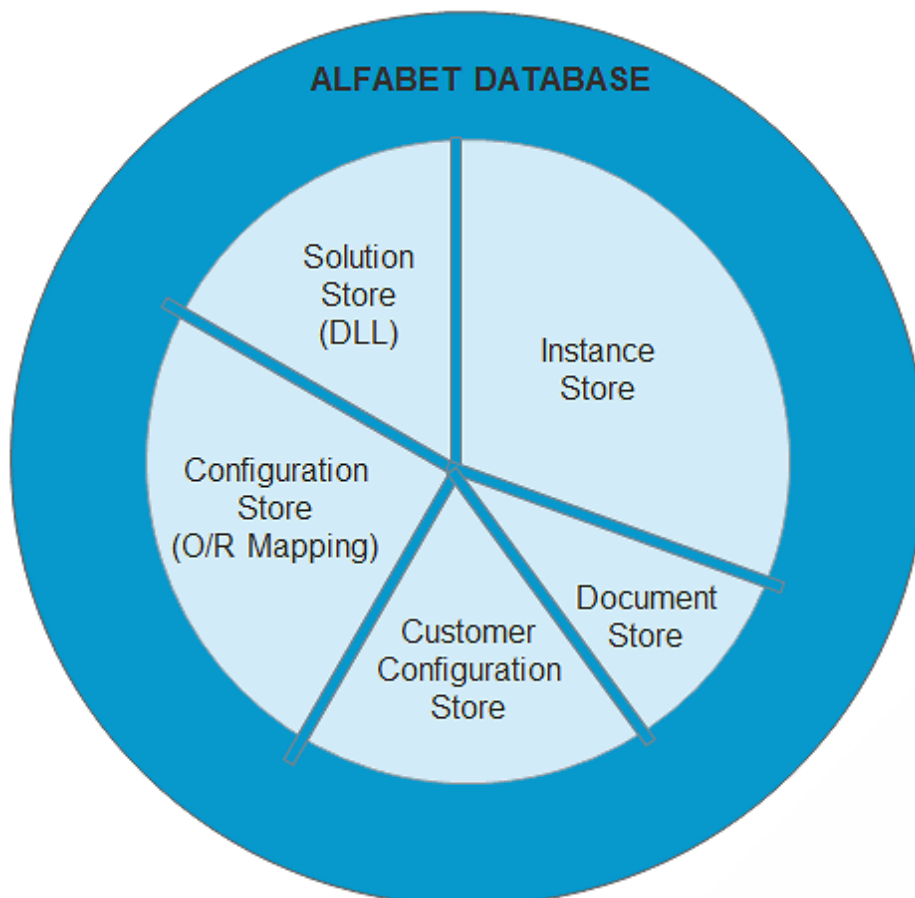
About Storage of Changes to the Meta-Model Performed with Alfabet Expand in the Alfabet database

Technically, changes to the meta-model may be performed via Alfabet Expand while at the same time:

- users are currently changing the data in the same database via a running Alfabet Web Application.

- the meta-model is changed by another solution designer in another instance of Alfabet Expand connected to the same database.

It is highly recommended to change the configuration via Alfabet Expand in a solution environment that is not connected to the production database to avoid that configuration changes interfere with users changing the data in the same database.

It is also highly recommended to change a database with one Alfabet Expand instance at a time.

Even if only one solution designer is changing the configuration in a development environment, it is useful to know how and when the configuration changes performed in Alfabet Expand are available in the Alfabet database and to the user connected to the Alfabet database via a running Alfabet Web Application. The solution designer might want to check the configuration by accessing the Alfabet user interface via a running Alfabet Web Application in the development environment.

Alfabet Expand Windows, Alfabet Expand Web and the Alfabet Web Application are loading a working copy of the configuration available in the Alfabet database at startup. The following applies for configuration changes:

- If a solution designer changes the configuration via Alfabet Expand Windows or Web, the changes are instantly applied to the working copy her/his Alfabet Expand instance, but they are only stored into the

    Alfabet database when the solution designer clicks the **Save** button in the toolbar of Alfabet Expand.

- The solution designer can clear changes that are performed in Alfabet Expand but not yet saved via the **Save** button using the button **Clear Changes**.This option re-loads the configuration from the Alfabet database into the local working copy of the configuration, overwriting any local changes.

- If a solution designer saves the configuration via the **Save** button, the configuration of the Alfabet database is changed, but changes are not instantly visible on an Alfabet user interface that is connected to the Alfabet database via a running Alfabet Web Application. The Alfabet Web Application also uses a working copy that is updated in regular intervals. Changes to the configuration of the Alfabet database might therefore only be visible to the user after several minutes. To ensure that the changes are visible in the user interface, the solution designer should re-start the web server hosting the Alfabet Web Application prior to checking new changes in the user interface. The Alfabet Web Application can be restarted directly from Alfabet Expand Web with the option **Meta-Model** > **Restart Web Application**.

    > The Alfabet Web Application can be configured to work in `Design` mode for testing purposes. One of the options available in design mode is a button **Meta-Model** in the user interface with options to re-read the meta-model or to re-start the web server. For more information, see *Special Configuration of Testing Environments* in the reference manual *System Administration*.

- The working version of the meta-model in Alfabet Expand is not updated automatically. Therefore, if one Alfabet Expand instance saves configuration changes to the database, the working version of the meta-model of all concurrently active Alfabet Expand instances is outdated. To update the working

version of Alfabet Expand with the current version of the meta-model configuration in the Alfabet database, click **Meta-Model** > **Reread Meta-Model** in the menu of your Alfabet Expand instance.

- Changes to object classes, like for example introducing a new custom property, are central changes that might interfere with the storage of object data that is stored when a user creates objects of the class. To ensure that the central class model a user is working with is always identical with the class model

  stored in the Alfabet database, saving a change to the object class model via the **Save** button will cause a semantic lock to the Alfabet database. All currently open connections from Alfabet components will be closed. A message will be displayed to solution designers concurrently working with other Alfabet Expand Windows instances, informing them about the lock to the database. Alfabet users currently working on the same Alfabet database in the Alfabet user interface or with Alfabet Expand Web are informed that their session expired. The meta-model changes will then be updated to the Alfabet database. The Web server must be restarted. Please note that updating the meta-model from a master database as described below in the section *Saving the Configuration of the Alfabet Solution to an AMM File* is considered a meta-model change regardless of whether the class model has been changed.

- A backup of the current configuration should always be made before making changes in order to prevent the inadvertent and irrevocable loss of important configuration data. An AMM file with the current configuration can be created via Alfabet Expand web to store the configuration prior to changing it.

Overview of Administrative Tasks Related to Solution Design

It is recommended that all changes to the solution configuration are performed in a development environment and tested in a test environment before the solution configuration is migrated to the production environment. All development and test environments should be based on copies of the production database. An overview of the development, test, and production environments is provided in the section *Best Practice Installation and Workflow* in the reference manual *System Administration*.

Software AG provides mechanism that allows the customer configuration specified in the development environment to be saved and restored in another database independent of other parts of the database. With these mechanisms, the configuration developed by a solution designer in a test environment can be first taken over to a test database and, after successful testing, to a production database without affecting the content of the instance store, which means the data created by users via the Alfabet user interface.

In general, two main mechanisms are available for taking over configurations from one database to another 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. Alfabet Expand offers a mechanism to create an AMM file storing the configuration from the development environment. Restore the configuration from an AMM file cannot be performed via Alfabet Expand. This task is usually performed in the production environment by a system administrator using the Alfabet Administrator. For taking over of the configuration from the development environment to the test environment, an AMM file can either be used by a system administrator using the Alfabet Administrator or by the tester having access to Alfabet Expand via the Alfabet Web Application of the test environment. Alfabet Expand also offers a mechanism to take over the configuration via a direct connection to the Alfabet Web Application of the development environment. The AMM file is then created on the fly and storing the configuration of the development environment and restoring it in the test environment is performed in a single action.

- Configuration relevant objects stored in the instance store of the Alfabet database can be imported to a target database via a direct connection between the source (master) database and the target database.

The target database must be configured to connect to the source database prior to using this functionality. The configuration can then be transferred via an administrator that is logged in to the Alfabet user interface and has access to the import functionality.

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 the production 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 and test database independent of the system administrator.

A typical solution design cycle includes the following database maintenance tasks:

- A new configuration is best carried out on an up-to-date copy of the production database. Backup files created during regular backup on the database level can be used to restore a copy of the production database in a test environment.

  The following is possible:

  - 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.

- After performing the required solution configurations, the solution designer can then save 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, for example by a system administrator using the tool Alfabet Administrator. Alternatively, a direct connection from the test environment to the development environment can be established that allows either the solution designer or the tester to update the test environment configuration using Alfabet Expand.

- Once tests have indicated that the configuration is correct, the configuration of the production database can be replaced or merged with the new configuration.
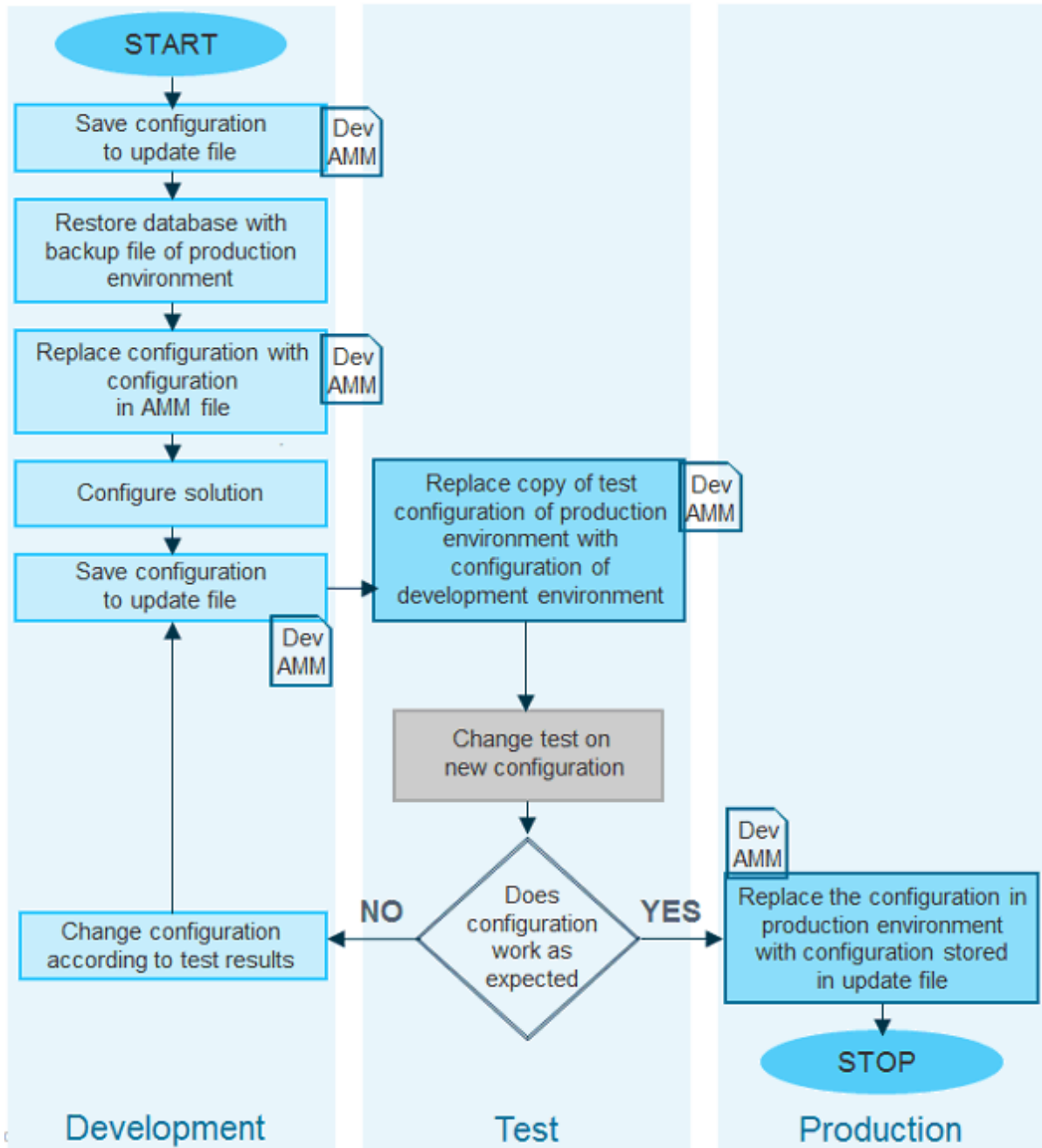
Whether configurations have to be merged or replaced with a new configuration depends on the configuration steps performed. The AMM file includes information about the update mechanism. It can exclusively be generated with the configuration tool Alfabet Expand because the solution designer can best decide on the update requirements to be specified for the AMM file.

FIGURE: Best practice for the configuration of the Alfabet solution

The figure above does not include the mechanisms for transferring configuration objects via a direct connection between databases. For the update via an AMM file from another Alfabet database in the test and production environment, an AMM based update via a direct connection can alternatively be performed. If required, the import of configuration objects in the instance store shall be executed afterward the update of the configuration via the AMM based mechanisms via the **Import Data Search** functionality.

Taking Over Configurations Performed in Alfabet Expand With AMM Based Mechanisms

Different AMM mechanisms are available that allow to take over the configuration, that means all configuration objects defined in Alfabet Expand and a subset of the reference data defined in the **Configuration** functionalities in the Alfabet user interface.

> ⚠️ Some configurations in Alfabet are not stored as part of the meta-model configuration and must be saved to a target database using other mechanisms. This applies to the following configurations:
>
> - **Configuration of some reference data configured in the Configuration module in the Alfabet user interface**: These configurations are saved in the instance store of the database. Only the selected subset of configurations listed above can be included in AMM files. The functionality **Import Data Search** can be used for import of the reference data that cannot be included into the AMM file. For more information,see *Importing Objects of Configuration Relevant Object Classes from a Master Database*.
>
> - **Mandate configuration**: The configuration of mandates is saved in the instance store of the database. The functionality **Import Data Search** can be used for import of mandate configurations. For more information, see *Importing Objects of Configuration Relevant Object Classes from a Master Database*.
>
> - **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 stylesheet files stored in the **Internal Document Selector** that are to be used in HTML templates implemented in the workflow capability. Although HTML templates can be saved to an AMM file and uploaded to a target database via the meta-model update, the images and stylesheet files must be uploaded to the **Internal Document Selector** of the target database via **Internal Documents** functionality in the Alfabet user interface, which is accessible via the `Admin` user profile.

During the creation of an AMM update file, the solution designer can decide about the scope of the configuration that shall be taken over to the target database. He/she can decide to select:

- The complete configuration.

- All configuration objects of selected types of configuration objects such as all workflows or all diagram view items.

- All configuration objects tagged with a defined tag that is set by the solution designer as part of the configuration process. It is additionally possible to store only tagged objects of selected configuration object types into the AMM file.

- A subset of configuration objects selected by the solution designer from a list of all available configuration objects according to his/her demand prior to creating the AMM file.

During the restore process, the parts of the configuration saved in the AMM file are merged with the corresponding configuration of the target database. The parts of the configuration that are not included in the AMM file are not affected by the merge action. Alternatively, the entire existing configuration in the target database can be replaced with the configuration in the AMM file. If this option is selected, the solution designer must make sure that the complete configuration is stored in the update file.

Alfabet Expand basically offers two mechanisms to take over configurations stored into AMM files to a target database:

- The AMM file is created by a solution designer from the development database in an Alfabet Expand connected to the development database. The solution designer can decide with maximum flexibility about which configuration objects to take over to the target database . The AMM file is then used to update the target database. This functionality is not available in Alfabet Expand Web. A system administrator must then use the AMM file to perform the update of the meta-model configuration in the target database via the Alfabet Administrator.

- The creation of the AMM file and the update of the configuration are both performed via an Alfabet Expand connected to the target database. In this case, a direct connection between the source and the target database must first be configured in the server alias of the target database. The complete update process including storage of data from the source database in the AMM file and update of the meta-model in the target database is performed on the target database, most likely by someone that did not change the configuration in the source database personally. Therefore, the selection mechanisms for taking over solution objects are restricted for this mechanism and the basic decision about overwriting the configuration in the target database or merging the configurations is defined in the server alias of the Alfabet Expand connecting to the target database.

For all AMM file based mechanisms, translations provided in the source database for the solution configuration objects stored in the AMM file are automatically also stored in the AMM file and added to the vocabulary of the target database during update of the meta-model with the AMM file. Vocabulary review information like proposed changes and reviewer information is not stored in the AMM file. The following columns of the **ALFA_SYS_VO-CABULARY** database table are included into the update via AMM file: ORIGINAL, SOURCE, ACCESSIBILITY, HASHED_ID, OBSOLETE, CREATE_DATE, OBSOLETE_DATE, and REPLACES.

The following information is available:

- Identifying Configuration Objects via Solution Tagging
  - Setting Tags for a Single Configuration Object
  - Setting or Removing Tags For Multiple Configuration Objects Simultaneously
  - Setting a Default Tag Automatically Applied to New and Changed Objects
- Versioning of Configurations
- Saving the Configuration of the Alfabet Solution to an AMM File
  - Saving Complete or Tagged Types of Configuration Objects or the Complete Configuration with Alfabet Expand
  - Saving Selected Objects of the Configuration with Alfabet Expand
    - Searching for Objects in the Find Results Table of the Meta-Model Objects Editor
  - Storing all Objects Tagged With a Selected Tag to an AMM File
- Updating the Configuration of the Alfabet Solution Environment with Alfabet Expand
- Direct Import of the Solution Configuration from a Master Database
  - Overwriting the Solution Configuration with the Configuration of a Master Database
  - Merging the Solution Configuration with the Configuration of a Master Database
  - Checking and Repairing the Configuration Updates From a Master Database
- Correcting Issues that Occured During Meta-Model Update
  - Checking the Success of Meta-Model Updates

- Securing and Checking Database Consistency with the Meta-Model

- Comparing Database Configurations

- Exporting Information About the Custom Configuration in an XML File

- Building Custom Reports that Inform About the Current Structure of the Standard And Custom Meta-Model

Identifying Configuration Objects via Solution Tagging

Solution objects can be marked with one or multiple tag names to define which solution configuration project the configuration is relevant for. This is useful to ease the selection of meta-model objects for deployment because you can search for a defined solution tag. The main reason for tagging, however, is to identify the objects that are part of a selected configuration when the configuration must be changed.

For example, a company-specific process requires the maintenance of custom attributes for an object class. The custom attributes defined for the process as well as the custom editor, the custom reports, and the workflow designed for the process are all tagged with the name of the process. After the solution is complete and successfully migrated to the production environment, an additional custom attribute was required as well as modification of the custom editor and the design of a completely different workflow.

During redesign in the development database, the solution designer is able to identify which objects might be affected by searching for the relevant solution tags. After changing the configuration, the production database must be updated. The solution designer selects all configuration objects tagged with the process name to upload to the AMM file.

During update of the production database, these objects will be updated. But in the case of the workflow template, the update is not sufficient because the old workflow template for the solution is not updated but deleted and replaced by a new workflow template with a different name. The solution designer can now configure the AMM file to trigger automatic deletion of all configuration objects tagged with the process name from the target database prior to writing the configuration stored in the AMM file to the database. In this way, the old workflow will be deleted from the target database and the new workflow template will be added.
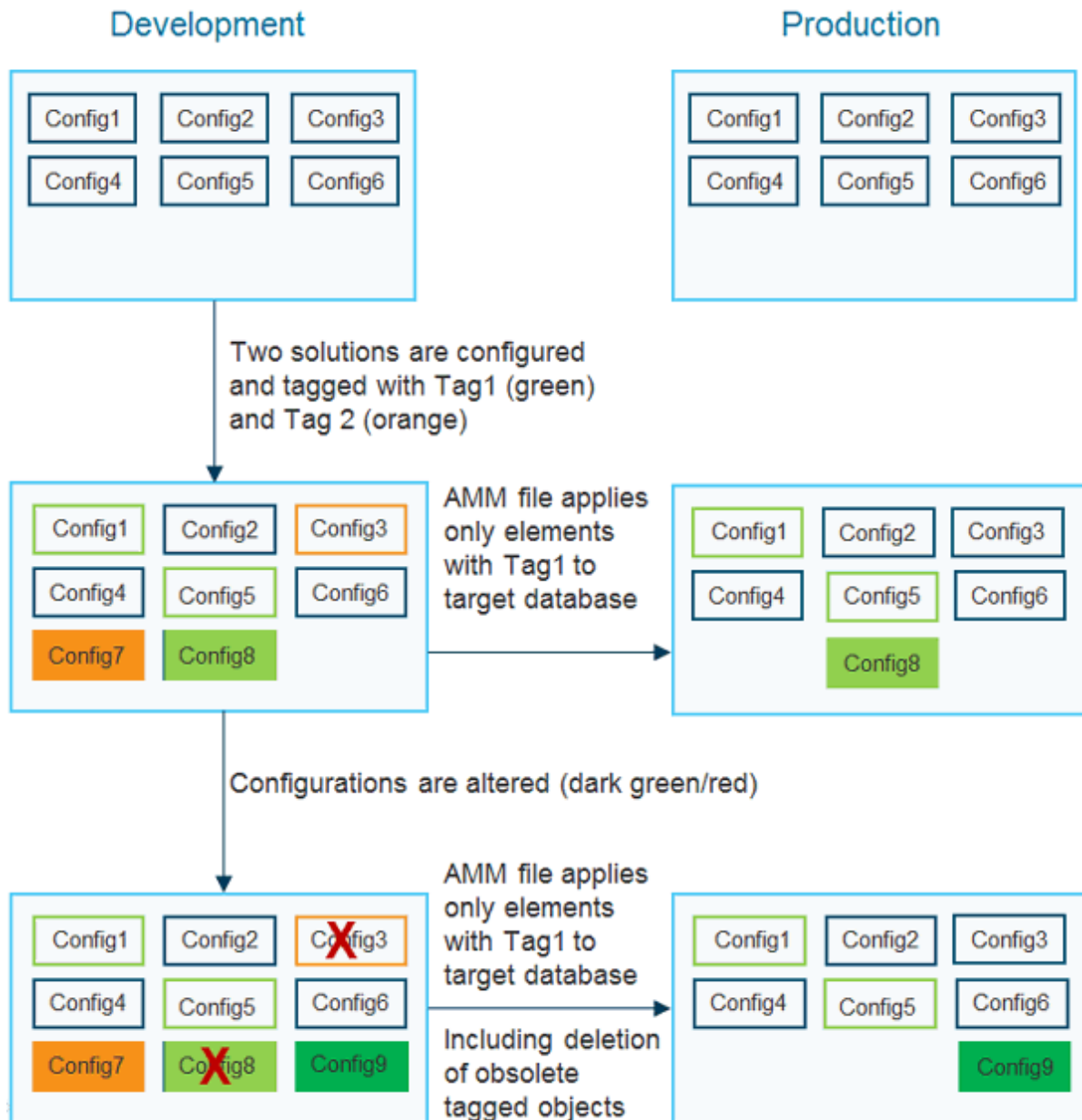
FIGURE: Configuration changes applied to a target database using solution tagging mechanisms in AMM files

> In addition to applying a tag to configuration objects, you can also alter the attribute **Version** of an object each time a reconfiguration is done to specify for which version of a feature this configuration is done. The **Version** attribute is part of the **Tech Info** specification for a configuration object.

Solution tags can be added to single objects while the object is configured or to a bundle of configuration objects during selection of the configuration objects for upload to an AMM file.

To avoid errors caused by misspelling solution tag names, the tag name will be defined only once the first time it is applied and is then selected from a selector for all subsequent tagging actions. The names of the solution tags are case-sensitive.

Instead of setting the tags for each individual configuration object or a bundle of configuration objects manually, you can configure Alfabet Expand to mark all new or changed configuration objects with a default tag. A solution designer can set the default tag to the task he/she is currently working with to ensure that all objects are correctly tagged and change the default tag when starting the configuration of the next feature. If you change an

existing configuration object that is tagged with a tag different to the selected default tag, the default tag is added to the list of tags for the configuration object. Existing tagging is not overwritten.

The default tag setting is removed when the user logs out from Alfabet Expand.

Once objects are tagged, you can create a AMM file including only the tagged objects via the mechanism described in the section *Saving the Configuration of the Alfabet Solution to an AMM File*.

If you want to delete obsolete tagged objects in the target database, make sure that the AMM file contains all objects currently tagged with the tag name are included in the AMM file and enter the name of the tag in the field **Provide comma-separated tags to find objects to be removed**. It is recommended that you use this mechanism if you want to only replace (rather than merge) the configuration of reports, workflows or ADIF schemes.

> ⚠️ You cannot remove changes to the class model (such as custom object classes and object class properties) with this mechanism.

Setting Tags for a Single Configuration Object

To tag a single configuration object:

1) Select the configuration object that you want to tag in the explorer.

2) In the attribute window of the configuration object, expand the **Tech Info** section.

3) Click into the field for the attribute **Tags** to open the **Meta-Model Tags** window.

4) If you want to add a new tag to the list, enter the name of the tag in the **Add New Tag** field and click the **New Tag** [✱] button.

5) Select one or more tags in the list to apply the tags to the configuration object.

6) Click **OK** to close the dialog.

Setting or Removing Tags For Multiple Configuration Objects Simultaneously

To simultaneously set a tag or remove tags from multiple configuration objects:

1) In the **Utilities** designer, click the **Meta-Model Deployment** node.

2) In the attribute pane, click the word `Config` in the field of the attribute **Meta-Model Update Content**. The **Configuration Update File Content Editor** opens.

3) All customer configured configuration objects are displayed in the table in the **Meta-Model Content** tab, sorted by category. The number of selectable objects in each category is displayed behind the category name. To view the objects in a category, expand the category by clicking the arrow in front of the category header. Alternatively, you can expand all categories by clicking the [2] in the first column header.

Configuration Update File Content Editor (PropGridMMDeployEditor)

Meta-Model Content | Guide Pages | Reference Data

Name		Creation Date Before		Last Update A
			📅	
Tags		Creation Date After		Last Update B
	▼		📅	

[Update]

📝 Set Tags   🔖 Remove Tags   ✕ Clear Tags   📑 Export ▼

	1	2	Name ▲	Type	Version	Date created	Da
1	▶		ADIFSchemes (116)				
118	▼		BusinessFunctions (219)				
119			AdHocReporting	AlfaBFunction	1.0	10/08/2009	3
120			ADMIN_AdifJobs	AlfaBFunction	1.0	21/06/2017	3
121			ADMIN_ArchiveManager	AlfaBFunction	1.0	11/06/2008	3
122			ADMIN_AutomatedTranslationReview	AlfaBFunction	1.0	17/04/2018	3

4) Select the objects that you want to tag in the table. You can select several objects in the table at the same time by holding down the CTRL key while selecting.

ⓘ Optionally, you can set the following filters and click Update to limit the display of objects in the table to objects matching a defined search condition:

- **Name**: Enter the name or part of the name of the configuration object or configuration objects that you are searching for. All configuration objects with a name including the given search string are displayed. A wildcard is not required for the search...

- **Tags**: Select one or multiple tags in the drop-down list to display only configuration objects that are tagged with one of the selected tags. Please note that filtering for a specific tag and then clearing the tag setting or removing the tag selected in the filter field from selected objects will immediately hide the objects from display in the table.

- **Creation Date Before**: Click the calendar icon on the right of the filter field and select a date in the calendar to limit display to configuration objects created before or at the selected date.

- **Creation Date After**: Click the calendar icon on the right of the filter field and select a date in the calendar to limit display to configuration objects created after or at the selected date.

- **Last Update After**: Click the calendar icon on the right of the filter field and select a date in the calendar to limit display to configuration objects last updated after or at the selected date.

- **Last Update Before**: Click the calendar icon on the right of the filter field and select a date in the calendar to limit display to configuration objects last updated before or at the selected date.

5) Do one of the following to alter the tag setting of the selected objects:

- To set one or multiple tags for all objects, click the **Set Tag(s)** option in the toolbar to open the **Meta-Model Tags** window. If you want to add a new tag to the list, enter the name of the tag in the **Add New Tag** field and click the **New Tag** button. The names of the solution tags are case-sensitive. Select one or more of the listed tags to apply the tags to the selected configuration objects and click **OK** to close the dialog.

- To remove selected tags from the selected objects, click the **Remove Tags** button in the toolbar. In the **Meta-Model Tags** window that opens, select the tags that you want to delete from the selected objects and click **OK** to save your changes.

- To remove all tags from the selected objects, click the **Clear Tags** button in the toolbar.

Setting a Default Tag Automatically Applied to New and Changed Objects

To set a default tag:

1) In the toolbar of Alfabet Expand, select **EXPAND DESIGNERS** > **Utilities**.
2) Click the **Utilities** root node in the explorer.
3) Click on the arrow on the right and select **Set Current Tag**.
4) If you want to add a new tag to the list, enter the name of the tag in the **Add New Tag** field and click the **New Tag** button.
5) In the **Select Tag** field, select the tag that you want to set as default tag to all new and changed configuration objects from the drop-down list.

> ⚠ The selection is removed when the user logs out of Alfabet Expand. If the selection shall be used again in the new session, the selection has to be repeated.

> ⓘ Within the current session, you can revert to working without a default tag by selecting the blank line on top of the drop-down list or by logging out and re-logging in to Alfabet Expand.

Versioning of Configurations

Optionally the solution designer can provide a name and version number for the configuration defined for the company. The configuration name and version defined in Alfabet Expand is written to all AMM files generated from the database and will overwrite a configuration name and version in the target database on update of the meta-model with the AMM files. the system administrator updating a database configuration with the AMM file can view the configuration name and version in the summary or the AMM content.

If a configuration name and version is defined, it will be displayed to users clicking **Help** > **About Alfabet** in the main menu of the Alfabet user interface on top of the Alfabet version information:

About Alfabet

Configuration Name 1.1
Alfabet 10.6.0

Alfabet Solution :
ITPlan v.10.6.0.0 (14/04/2020)

To define a configuration name and version:

1) In Alfabet Expand, open the **Utilities** designer.

2) In the explorer, click the **Utilities** node.

3) In the attribute window, define the name and version for the configuration in the **Configuration Name** and **Configuration Version** attributes.

4) Click the **Save** button to save your changes.

Two new **Configuration Name** and **Configuration Version** attributes have been added to the **Utilities** node in Alfabet Expand Web. These attributes have already been available in Alfabet Expand Windows in the **Environment** node in the **Admin** tab. The name and version defined with these attributes will be written to all AMM files generated from the database and will be overwritten in the target database with the values in the AMM file on update of the meta-model. Solution designers can optionally use these attributes to manage the version history of changes applied to a production environment. The **Configuration Name** and **Configuration Version** values are shown in the **About Alfabet** screen.

Saving the Configuration of the Alfabet Solution to an AMM File

You can save the customized meta-model configuration of 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 how 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 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.

Restore or merge of a configuration in an AMM file to a target database can only be performed with the Alfabet Administrator. For more information, see the section *Restoring the Configuration of the Alfabet Solution Environment* in the reference manual *System Administration*.

Storing All or Selected Parts of the Configuration in an AMM File

After having performed a solution configuration, you can select the solution configuration objects that you have altered and add them to an AMM file. The AMM file created in Alfabet Expand will merge the configuration into the target database. Replacement of the configuration is not configurable.

Guide pages, reference data and user profiles that are stored in the database can be selected as well as the configuration objects changed in Alfabet Expand:

The following objects in the configuration can be selected to upload to an AMM file:

Designer in Alfabet Expand	Configuration Object (Section name in Selector for AMM file)
Class Designer	- Object Class (`Classes`) - Enumeration (`Enumerations`) - Database View (`DbViews`) - API Cultures (`APICultures`)
Presentation Model Designer	- Object View (`ObjectViews`) ⚠️ If you select an object view, the object view is created in the target database without workspaces. The workspace objects of the object view must also be selected to be included in the AMM file. Please note that existing workspaces with the same name will be overwritten. ⚠️ If you select an object view, the object cockpits defined for the object view will be included in the AMM file. If an object view with the same name exists in the target database, the object cockpits of the object view in the target database will be deleted and replaced with the object cockpits of the object view in the AMM file. ⚠️ All configured reports added to workspaces or to an object cockpit of the object view must also be selected. If a custom help is defined to a view in the object view, this view must be selected separately. - Workspace, subordinate object of Object View (`Workspaces`) ⚠️ If you select a workspace, the parent object view as well as all associated workspaces that should be present in the final configuration of the target database must also be selected. If the workspace includes standard page views with a custom help definition or configured reports, these must also be selected. - Standard Page Views with a custom help definition (`Views`)

Designer in Alfabet Expand	Configuration Object (Section name in Selector for AMM file)
	• Selector (`Selectors`)
	• Custom Editor (`CustomEditors`)
	• Editor (`EditView`)
	ⓘ An editor is added to the configuration in the context of a survey. While a custom editor is a customer defined editor tab appended to a standard editor of a standard object class, an editor listed in this section is defined for a customer defined survey class with no standard editor. Surveys are not editable in Alfabet Expand Web.
	• Wizard (`Wizards`)
	• View Scheme (`ViewSchemes`)
	• Class Setting (`ClassSettings`)
	• Monitor Template (`Monitors`)
	• XMLObject (`XMLObjects`)
	• Solution Manager (`SolutionManagers`)
	• Text Template (`TextTemplates`)
	• GUI Scheme (`GuiSchemes`)
	ⓘ A custom logo defined in a GUI Scheme is automatically included as well.
	• Visualization Item (`VisualizationItems`)
	⚠ If the HTML templates in the Visualization Items folder contain links to images or stylesheets stored in the **Internal Document Selector**, the images and stylesheets are not stored in the AMM file. The content in the **Internal Document Selector** must be manually uploaded to the **Internal Document Selector** of the target database after updating the target database with the configuration in the AMM file.
Icon Gallery	• Icon Group (`IconGroups`)
	ⓘ If the icon group contains icons that you uploaded to the Alfabet database, you must also select the `Icons` in the section Icons in the table or later select the option **Save Icons** of the AMM file to upload all icons in the current database.
	• Icons (`Icons`)

Designer in Alfabet Expand	Configuration Object (Section name in Selector for AMM file)
Business Function Designer	• Custom Explorer (`BusinessFunctions`) • Standard Business Functions with a custom help definition (`BusinessFunctions`)
Workflow Designer	• WorkflowTemplate ⓘ The structuring of workflows in workflow template groups is saved in the workflow template object with the attribute **Group**. If you select a workflow template that is displayed in a workflow template group that is not present in the target database, the workflow template group will automatically be created in the target database and the workflow template will be displayed under the group node in the explorer.
Reports Designer	• Report Folder (`ReportFolders`) ⓘ If you select a report folder, the folder will be saved to the AMM file and created in a target database updated with the AMM file without content. The reports in the report folder must be added separately to the AMM file. • Report (`Reports`) ⓘ If you select a report located in a report folder that is not present in the target database updated with the AMM file and also not present in the AMM file containing the solution, the report will be saved in the **Reports** root folder in the target database. If the report is located in the sub-folder of a report-folder, all report folders in the hierarchy must be present either in the target database or in the AMM file in order to save the report in the same folder hierarchy as in the source database.
Publication Designer	• Publication (`Publications`)
ADIF Designer	• ADIF Import Scheme and ADIF Export Scheme (`ADIFSchemes`) ⓘ The structuring of ADIF schemes in groups is saved in the ADIF scheme object with the attribute **Group**. When you select an ADIF scheme displayed in a group that is not present in the target database updated with the solution, the group will be automatically created in the target database and the ADIF scheme is displayed under the group node in the explorer.

Designer in Alfabet Expand	Configuration Object (Section name in Selector for AMM file)
	ⓘ After update of a target database with the AMM file, all ADIF import schemes that are available in the target database after the update and have the attribute **Auto Run** set to `True` are automatically executed.
Diagram Model Designer	• custom Diagram Definitions (`DiagramDefinitions`) • Custom Diagram Item Template (`DiagramItemTemplates`) ⓘ If a custom diagram item template is configured to display icons, the icons must be separately selected for upload. Icons are not automatically included as sub-objects of the custom diagram item template.
Reusable Elements	• Event Template (`EventTemplates`) • Resource Bundle (`ResourceBundles`)

To create an AMM file that can be used to update the meta-model of another Alfabet database via the **Update Meta-Model** functionality of the Alfabet Administrator:

1) Open the **Utilities** Designer.

2) Click the **Meta-Model Deployment** node.

3) If you have used the functionality before, optionally clear the previous selection by clicking the arrow on the right of the selection bar and selecting **Reset Deployment**. All previous settings are removed from the **Meta-Model Deployment** node.

4) Set the following attributes of the **Meta-Model Deployment** node:

- **Name**: Enter a meaningful name for the meta-model deployment. 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.

- **Description**: Enter a meaningful description that explains the purpose of the update in the target database. 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.

- **Migrate Workflows**: Select the checkbox in the **Meta-Model Content** tab if you have included workflow template configuration in the `AMM` file and 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*.

- **Remove All Guide Pages**: select the 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.

- **Remove Tags**: 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 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*.

  ⚠ 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.

5) In the **Meta-Model Update Content** attribute, click the word `Config` in the field.
   The **Configuration Update File Content Editor** opens.

6) In the **Meta-Model Content** tab, all available solution configuration objects that are configured in Alfabet Expand are displayed in an expandable table, sorted by category. To limit the display of objects in the table, you can set the following filters and click **Update**:

- **Name**: Enter a search string. Display is limited to configuration objects with a name containing the search string or identical to the search string.

- **Tags**: Select one or multiple solution tags from the dropdown-list. Display is limited to configuration objects tagged with at least one of the selected tags.

- **Creation Date Before**: Select a date from the calendar. Display is limited to configuration objects created at or before the selected date.

- **Creation Date After**: Select a date from the calendar. Display is limited to configuration objects created at or after the selected date.

- **Last Update Before**: Select a date from the calendar. Display is limited to configuration objects last updated at or before the selected date.

- **Last Update After**: Select a date from the calendar. Display is limited to configuration objects last updated at or after the selected date.

7) Select the objects that you want to add to the AMM file in the table.

  ℹ Note the following about working with the table in the Meta-Model tab:

  - The number of selectable objects in each category is displayed in parentheses behind the category name.

  - To view the objects in a category, expand the category by clicking the arrow in front of the category header. Alternatively, you can expand all categories by clicking the ▣ in the first column header.

  - You can select several objects in the table at the same time by holding down the CTRL key while selecting.

  - If too many results are found, the results are displayed on multiple pages. If you click into the table, a floating toolbar is displayed that informs you about the number of pages available and provides navigation within the pages.

  - The table displays the following information about the configuration object:

- **Name**: The value of the **Name** attribute of the configuration object.

- **Type**: The type of configuration object in the Alfabet meta-model. When exporting the information about the meta-model to an XML file with the functionality **Meta-Model** > **Save Configuration** of Alfabet Expand, information about the object is stored in an XML element with the name given as **Type** here.

- **Version**: The value of the **Tech Info** > **Version** attribute of the configuration object.

- **Date created**: The value of the **Tech Info** > **Creation Date** attribute of the configuration object.

- **Date modified**: The value of the **Tech Info** > **Last Update** attribute of the configuration object.

- **Protection Level** : `Public` objects are created by the customer, `Protected` objects are standard configuration objects of the Alfabet meta-model that may have been modified by the customer.

- **Tags**: The value of the **Tech Info** > **Tags** attribute of the configuration object. Solution tagging is designed to mark all configuration objects that are required for a specific functionality with a tag to ease the identification of relevant objects. For more information, see *Identifying Configuration Objects via Solution Tagging*.

- **Notes**: The value of the **Tech Info** > **Implementation Notes** attribute of the configuration object.

- The results of the search displayed in the table can be exported to a Microsoft® Excel® file to provide the information about selection of objects to persons that do not have access to Alfabet Expand or to archive the content of an AMM file in a readable format. To save the content of table, click the **Export** button in the toolbar of the table and select the output format. The file is then created and provided for download in a ZIP file. Select the option **Include Filter Summary** to include information about the current filter settings.

8) A selection container is displayed below the table. In the toolbar above the selection container, click the **Add** button. The objects which you selected in the table are added to the selection container. To remove a subset of the selected objects from the selection container, select the objects in the selection container and click the **Remove** button in the toolbar of the selection container. To remove all objects from the selection container, click the **Remove All** button.

> It is recommended that you tag the selected objects with a solution tag prior to adding them to the selection container. Tagging allows objects in both the source and the target database to be identified.

> You can save the configuration in the selection container in a Microsoft® Excel®, Microsoft® Power Point® or HTML file. This feature may be useful to generate reports about the configuration update, compare different stages of configuration, or to archive different stages of configuration. In the toolbar of the selection container, click the **Export** button in the toolbar of the selection container and select the output format. The file is then created and provided for download in a ZIP file.

9) 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 into the existing guide page configuration of the target

database. To save all guide page projects displayed in the table to the AMM file, click the **Check All** button. Click the **Clear All** button to clear the selection of all guide page projects in the table.

> ⚠ Please note that you must include all images in the configuration of the guide pages by selecting them in the **Meta-Model Content** tab.

> ⚠ 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 AMM into 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*.

10) Optionally, go to the **Reference Data** tab and select user profiles and a specified subset of configurations performed in the **Configuration** module in Alfabet to be uploaded. 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. In contrast to solution configurations performed in Alfabet Expand, solution objects stored in the instance store are not overwriting the instance store tables in the target database. Instead, the configuration is merged.

> ⓘ 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
PrioritizationScheme	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 **Clear 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 into 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.

11) After having selected all relevant configuration objects, click **OK** to close the selection container.

12) Click the arrow on the right of the selection bar of the **Meta-Model Deployment** node and select **Create Deployment**.

13) In the window that opens, click the **Download** button to download the AMM file and store it to your local file system.

Updating the Configuration of the Alfabet Solution Environment with 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*.

- 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 databse 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. This check can be de-activated on performing the update of the meta-model. Please note that de-activation may lead to problems with strings displayed as original being defined in different languages.

  - 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 **Utilities** nodein the **Utilities** Designer. 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

  - User Profiles

  - Data Quality Rule Groups

  - Data Quality Rules

  - Business Question Groups

  - Business Questions

  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.

(i) If the functionality for update of the meta-model is not working, please check the alias configuration of Alfabet Expand. The option **File-Based Update Permitted** must be activated in the **Expand** tab of the alias configuration editor. This option is activated by default.

Prior to updating the meta-model, ensure that no Alfabet components are currently connected to the Alfabet database and that replication mechanisms that target the Alfabet database are shut down.

1) Open the **Utilities** designer.

2) Click the **Meta-Model Configuration** node in the explorer.

3) Click the arrow at the right of the selection bar and select **Update Meta-Model Configuration** > **Case Sensitivity and Configuration Culture Check**.

   (i) A case-sensitivity check has been implemented for update of the meta-model via AMM file. If the target database is case-insensitive and the AMM 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. If the AMM file contains configuration objects differing only in the use of upper case and lower case letters, these configuration objects are regarded as different configuration objects in a case-sensitive database while they are regarded as the same configuration object in a case insensitive database.

   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. If you merge the configuration of databases with different configuration languages, the original configuration of the target database will contain configurations in two different languages which leads to translation problems.

   Both checks can be deactivated by using the following options instead of the **Case Sensitivity and Configuration Culture Check**:

- **No Case-Sensitivity Check**: Only the check for configuration language compatibility is performed. If the case-sensitivity settings of the AMM file configuration and the target database are incompatible, this option should only be selected if the AMM file contains only configuration objects differing in more than the upper and lower case.

- **No Configuration Culture Check**: Only the check for case-sensitivity is performed. If the AMM file includes the culture settings from the source database, the **Primary Culture** settings will be taken over to the target database but the **Configuration Culture** setting will not be changed. Deactivating the check can lead to translation issues because it will merge configuration objects defined in another than the current configuration language to the target database.

- **No Case-Sensitivity and No Configuration Culture Check**: No compatibility checks are performed. If the AMM file includes the culture settings from the source database, the **Primary Culture** settings will be taken over to the target database but the **Configuration Culture** setting will not be changed. Deactivating the check can lead to translation issues if the AMM file and the target database have different configuration cultures and it can corrupt the database. If the case-sensitivity settings of the AMM file configuration and the target database are incompatible, this option should only be selected if the AMM file contains only configuration objects differing in more than the upper and lower case.

4) In the **Upload File** dialog, click the **Select File** button and select the AMM file from the local file system.

5) Click the **Upload** button. The AMM file is sent to the Alfabet Server to perform the update of the meta-model. During update of the meta-model, all database connections are closed. This also includes the connection from Alfabet Expand.

After applying a configuration to an Alfabet database, it is recommended that you review the Alfabet database for consistency with the meta-model as described in the section *Securing and Checking Database Consistency with the Meta-Model*.

Direct Import of the Solution Configuration from a Master Database

This functionality is only available if the server alias used to connect with Alfabet Expand to the target database is configured to establish a connection to the source database. For information about the required configuration, see the chapter Installation.

The server alias configuration also includes a setting that decides about the update method. Configurations can either be taken over completely, which means that the configuration of the target database is overwritten, or the configuration of the source database can be merged into the configuration of the target database. In the latter case, the user triggering the configuration update with Alfabet Expand on the target database can decide about which part of the configuration shall be included in the update.

For the update of the solution configuration from a master database, a restore mechanism is available. The history of performed updates can be displayed and logging information can be checked for each import. If the import has failed, the configuration of any point in the update history can be restored in the target database.

Please note the following about the process to update the meta-model from a master database:

- During update of the meta-model 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*.

⚠ **Before you update a production database, you must backup the target database for security reasons.**

Update of the meta-model from a master database 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 from master database 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 taken over via master database update. 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 into 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 like role types and cost types or indicator types and 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 into 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 into 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.

The following information is available:

- [Overwriting the Solution Configuration with the Configuration of a Master Database](#)

- [Merging the Solution Configuration with the Configuration of a Master Database](#)

- [Checking and Repairing the Configuration Updates From a Master Database](#)

Overwriting the Solution Configuration with the Configuration of a Master Database

Complete updates will be performed if the update of the configuration from a master database is triggered with a server alias of Alfabet Expand that is configured to perform complete updates.

The method is recommended to take over configurations from a development to a test environment after complete performance of a configuration. The tester can take over the configuration without any knowledge about configuration details.

Do the following to overwrite the configuration in your current database with the configuration in a master database:

1) Open the **Utilities** designer.

2) Click the **Meta-Model Configuration** node in the explorer.

3) Click the arrow at the right of the selection bar and select **Update Meta-Model Configuration from Master Database**. The **Update Configuration** dialog opens.

4) In the **General** tab, define the following fields:

- **Name**: Enter a meaningful name for the update. The name is used in the update history to identify the update.

- **Description**: Optionally a description that is displayed in the update history to provide more information about the update.

5) Optionally, go to the **Reference Data** tab and select user profiles and a specified subset of configurations performed in the **Configuration** module in Alfabet to be uploaded. 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. In contrast to solution configurations performed in Alfabet Expand, solution objects stored in the instance store are not overwriting the instance store tables in the target database. Instead, the configuration is merged.

> (i) 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.
ClassConfigu-ration	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:

Update Configuration (MMUpdateFileCreator)

General | Guide Pages | Meta-Model Content | **Reference Data** | Assemblies

Select Class

Evaluation Types ▼

Check All	Clear All	Checked Dependent Objects

	Name ⌃
1	Miscellaneous
2	Action
3	Adaptability (local)
4	Agreement of Service Level
5	Alignment with approved Digital Business Platform
6	Amount of Loss
7	API Assessment
8	Application Budget
9	Application Mode Strategy
10	Application Overall Riskiness
11	Application Processing Location
12	Application Protection Requirements
13	Application User Satisfaction
14	Application VVZ DQ Score

OK

> To save all configuration objects or user profiles currently displayed in the table to the AMM file, click the **Check All**button. Click the **Clear 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 into 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.

6) In the tab **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. The functionality on this tab is limited to upload of assemblies already available in the master database.

> For an overview about assemblies and their role in configuration, see the section *Managing Assemblies*.

7) Click **OK** to create the temporary AMM file. The update of the meta-model of the current database is started automatically without user interaction required.

8) Confirm the message by clicking **OK**. All connections to the Alfabet database are closed during update.

Merging the Solution Configuration with the Configuration of a Master Database

Updates of selected parts of the configuration will be performed if the update of the configuration from a master database is triggered with a server alias of Alfabet Expand that is configured to perform selective updates.

The method is recommended to take over defined parts of the configurations from a development to a test environment. A solution designer can for example tag all configurations for a defined feature with a specific test and the tester can be advised to take over all configuration objects with that tag. Additional, incomplete configurations that are currently available in the master database are not taken over to the test environment during the selective update, because they are tagged differently.

Selective updates are highly flexible. During the update, the person performing the update decides whether to take over

- the complete configuration,

- only selected types of configuration objects, like for example configured reports only, or

- only objects tagged with a defined tag.

It is also both possible to overwrite the existing configuration or to merge the configuration from the master database into the existing configuration of the target database.

Do the following to take over selected parts of the configuration in a master database to your current database:

1) Open the **Utilities** designer.

2) Click the **Meta-Model Configuration** node in the explorer.

3) Click the arrow at the right of the selection bar and select **Update Meta-Model Configuration from Master Database With Case-Sensitivity Check**. The **Update Configuration** dialog opens.

> (i) A case-sensitivity check has been implemented for update of the meta-model. If the target database is case-insensitive and the master database is case-sensitive, the update process will not be performed and a message informs about the incompatibility of the databases. You can use the **Update Meta-Model Configuration from Master Database with No Case-Sensitivity Check** option to update the meta-model of a case-insensitive database from a master database with a case-sensitive database. This option should only be used if the master database does not contain configuration objects differing only in the use of upper case and lower case letters. These configuration objects are regarded as different configuration objects in a case-sensitive database while they are regarded as the same configuration object in a case insensitive database.

4) In the **General** tab, define the following fields:

- **Name**: Enter a meaningful name for the update. The name is used in the update history to identify the update.

- **Description**: Optionally a description that is displayed in the update history to provide more information about the update.

5) 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.

> ⓘ 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.

> ⓘ 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 event templates defined in the **Reusable Elements** tab in Alfabet Expand.

- **Save Resource Bundles**: Select the checkbox to save the resource bundles for the generic Open API integration interface. Resource bundles are defined in the **Reusable Elements** 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 Data Workbenches**: Select the checkbox to save the content of the **Data Workbenches** tab in Alfabet Expand

- **Save Database Views**: Select the checkbox to save the database views defined in the **Meta-Model** tab in Alfabet Expand.

> ⓘ 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.

> 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.

> 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.

6) Select the following options, if applicable for your configuration:

- **Include Configuration Name and Version**: Deselect the checkbox if you do not want the configuration name and version defined in the **General** tab to be stored as configuration name and version in the meta-model of the target database.

- **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 visible 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 **Publication-Reports** 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 selected 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 objects shall be deleted in the target database prior to import of the configuration objects in the AMM file.

7) 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. You can use the Check All button to select all guide page projects in the table or the **Clear All** button to clear the selection of all guide page projects in the table.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 selected 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*.

8) In the **Reference Data** tab, user profiles, default views defined for data workbenches, 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.

ℹ️ 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.
`ClassConfigu-ration`	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:

Update Configuration (MMUpdateFileCreator)

General | Guide Pages | Meta-Model Content | Reference Data | Assemblies

Select Class

Evaluation Types ▼

	Name ⌄	Save
1	Miscellaneous	☑
2	Action	☐
3	Adaptability (local)	☐
4	Agreement of Service Level	☐
5	Alignment with approved Digital Business Platform	☐
6	Amount of Loss	☑
7	API Assessment	☐
8	Application Budget	☐
9	Application Mode Strategy	☐
10	Application Overall Riskiness	☐
11	Application Processing Location	☐
12	Application Protection Requirements	☐
13	Application User Satisfaction	☐
14	Application VVZ DQ Score	☐

Check All   Clear All   Checked Dependent Objects

OK   Cancel

(i) To save all configuration objects or user profiles currently displayed in the table to the AMM file, click the **Check All** button. Click the **Clear 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.

9) 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.

10) Click **OK** to create the temporary AMM file. The update of the meta-model of the current database is started automatically without user interaction required.

11) Confirm the message by clicking **OK**. All connections to the Alfabet database are closed during update.

Checking and Repairing the Configuration Updates From a Master Database

A history is available for configuration updates from a master database. The history also provides access to log information for the update file and a summary of configuration objects updated.

ⓘ The history only lists meta-model updates from a master database and data anonymization actions. File based updates of the meta-model are not included in the history.

Every entry in the history can serve as a restore point to re-establish the configuration prior to the selected update in the history.

ⓘ If update from a master database is not enabled, the history is nevertheless displayed to inform about the anonymization actions. Restore of the configuration from a restore-point is then deactivated.

Do the following to check and optionally restore the configuration:

1) Open the **Utilities** designer.

2) Click the **Meta-Model Configuration** node in the explorer.

3) Click the arrow at the right of the selection bar and select **Show Meta-Model Configuration History**. The history opens in a new window.

4) The history shows the following information:

> ⓘ You might need to scroll from left to right or enlarge the window to see all relevant information.

- **Name**: The name of the update defined during triggering of the update in the update dialog window.

- **Description**: The name of the update defined during triggering of the update in the update dialog window.

- **Update Time**: The date and time when the update was performed.

- **Executor**: The first name and name of the user that triggered the update process.

- **Product Version**: The version of the software that was imported from the source database. Please note that the version must match the current version of the target database. You can for example not restore the configuration from an update with a different version number than the current one.

- **Update Status**: The update status should be `Finished`. If it is `Error`, the update failed and you should check the log file for more information about the error that occurred. If the update process led to problems in your configuration, because only partial updates were performed, you can restore the configuration prior to the update by clicking the update in the table and clicking **Restore Configuration** in the toolbar. If the update was not performed at all, the log file is empty.

5) You can do any of the following:

- **To view the content summary of the AMM file** an update is based on, select the update in the table and click **Show Summary** in the toolbar.

- T **o view the log file of the update process**, select the update in the table and click **Show Update Log** in the toolbar. The log file includes the content summary of the AMM file and informs about the success of the update steps.

  > ⓘ Log messages are displayed in the language currently set for the master database.

- **To export the content of the history table** click **Export** in the toolbar. Select the output format from the drop-down list. A new window opens. Click the **Download** button to download the document.

- `To restore the configuration prior to an update`, click the update that you want to undo, and click **Restore Configuration** in the toolbar. Confirm the warning by clicking **Yes**. The restore is executed asynchronously.The restore may take some time. All connections to the database are closed during update.

  > ⚠ Please note the following:
  >
  > - The current version of the Alfabet database must match the version number of the restore point to perform the restore.

- Reference data configuration in the instance store, like for example cost types, indicator types or user profiles, is not restored via this mechanism.

Building Custom Reports that Inform About the Current Structure of the Standard And Custom Meta-Model

Importing Objects of Configuration Relevant Object Classes from a Master Database

Configurations may require the creation of objects for object classes in the Alfabet meta-model. For example:

- User profiles

- Mandates

- Reference data configured in the Configuration functionalities in the Alfabet user interface, for example, role types or cost types.

- Monitors

- Configured Reports

The AMM update mechanism only includes a subset of the objects stored in the instance store.

For import of other parts of the configuration objects stored as objects of object classes in the Alfabet meta-model, a unidirectional, direct connection can be established between two Alfabet database s for data update in the target database with data in the source database. Once the target database is configured to connect to the source database, a user can perform the import of new configuration relevant meta-model objects via the functionality **Import Data Search** of the Alfabet user interface. The import mechanism informs the user whether objects were added, and new objects can then be selected for import.

Optionally, the import mechanism can be configured to also detect changes to given configuration objects. This option is only available for a subset of the relevant classes.

⚠ The import mechanism has been developed for configuration objects only and shall not be used for import of any other object classes. The following limitations apply:

- The import mechanism matches the objects in the source and target database exclusively on basis of the **Image Properties** defined for the object class in the relevant class settings. Image properties must be carefully set to ensure that they can be used to unambiguously identify single objects. Source database objects with image properties not identical to the image properties of an object in the target database are regarded as new objects.

- The import mechanism optionally detects changed objects by comparing the LAST_UPDATE property of the object in the source and target database. If the LAST_UPDATE date of the object in the source database is younger than the LAST_UPDATE date of object in the target database, the object is marked as changed. Please note the following:

  - The comparison is based on the date without time. Therefore, if an object is for example imported and then changed in the source database on the same day, the object is not regarded as changed when performing the next update.

  - The way the object has been changed is not considered by this mechanism.

- This feature is not available for all relevant object classes and may require additional configuration of the object class.

- During import, the object is created in the target database or, if an object with identical values for the image properties already exists in the target database, the object in the target database is overwritten. A merging process is not available.

- Values for object class properties of the type `ReferenceArray` are ignored during import.

- Values for object class properties of the type `Reference` are only imported if the referenced object already exists in the target database. If you would for example import a hierarchy of cost types, you must first import all root type cost types and then all cost types of the second level of the hierarchy followed by all cost types of the third level of the hierarchy. If you import a cost type on the second level of the hierarchy without prior import of the parent cost type, the cost type will become a root level cost type in the target database.

- Values for object class properties of the **Type** `String` that are based on an enumeration, that means that the **Type Info** attribute of the property is set to the name of an enumeration, are only imported if the specified enumeration item is available in the target database. It is recommended to import the enumeration configuration of the source database into the target database via AMM file based meta-model update prior to using **Import Data Search**.

The following information is available:

- [Configuring the Connection Between the Source and Target Database](#)

  - [Configuring the Source Database to Provide Access to the Relevant Data via the Alfabet RESTful Services](#)

  - [Configuring the XML Object AlfabetIntegrationConfig of the Target Database](#)

  - [Creating an Alfabet Database Connection in the Target Database](#)

  - [Providing Access to the Import Data Search Functionality](#)

  - [Optional Configuration for Detection of Changes to Configuration Object](#)

- [Importing Configuration Objects from a Source Database](#)

Configuring the Connection Between the Source and Target Database

The connection to the source database is based on the Alfabet RESTful services. Establishing the connection requires that two Alfabet Web Application s are running, one with a connection to the source and one with a connection to the target database. The release version of the Alfabet components must be identical for source and target database. When a user triggers the data import mechanism on the Alfabet user interface, the Alfabet Web Application of the target database sends RESTful service requests to the Alfabet RESTful API of the Alfabet Web Application of the source database to request the data.
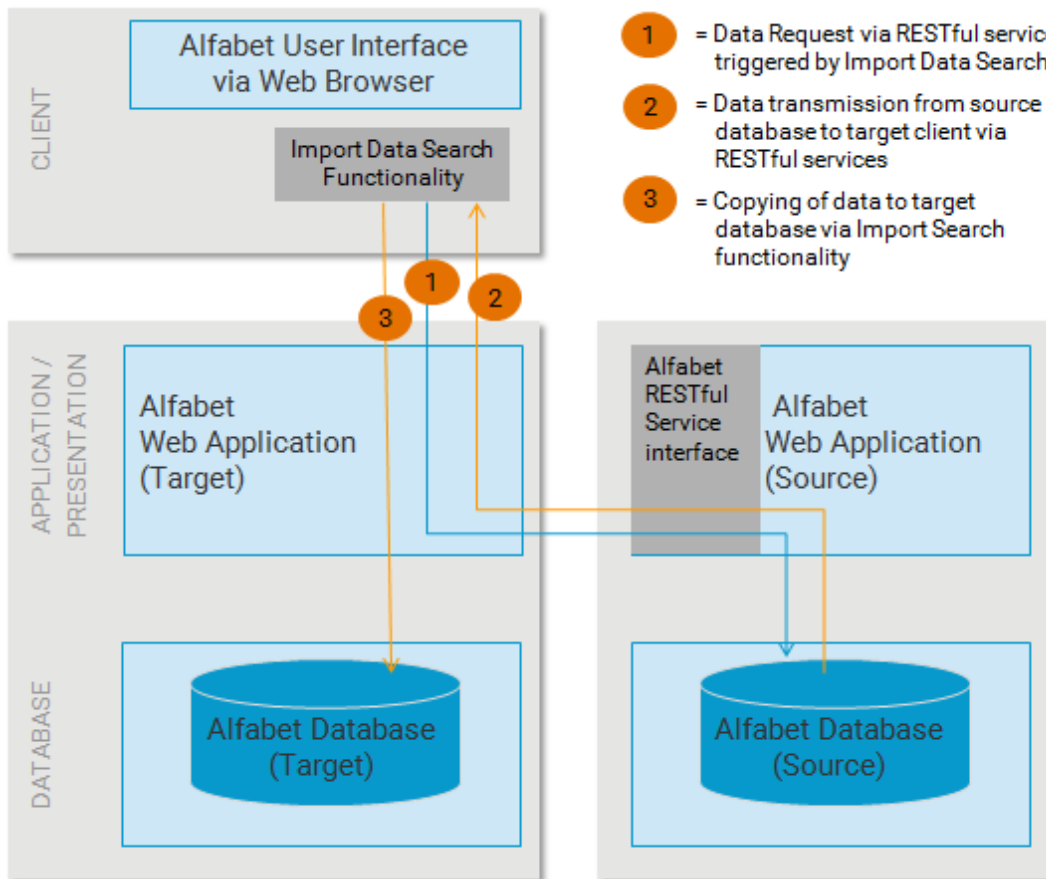
The configuration includes the following:

- Configuration of the Alfabet Web Application of the source database to activate the Alfabet RESTful API and set the required access permissions for access to the relevant data in the source database.

- Configuration of the Alfabet Web Application of the target database to act as a RESTful client. This configuration is done via Alfabet Expand in the XML object `AlfabetIntegrationConfig`. The configuration also includes a definition of the object classes that shall be imported.

- Configuration of the user access to the functionality in the target database. For this purpose, an object of the object class Alfabet Database Connection must be created via the Alfabet user interface and the access to the Alfabet database connection as well as to objects of the relevant object class must be granted to a user via the user profile assigned to the user.

The definition of object classes for that import is performed is typically further limited with each configuration step:

- Typically, only one database connection is configured in the XML object **AlfabetIntegrationConfig** for the connection to the source database. The configuration includes the definition of all object classes that shall be imported from the source database.

- Multiple different Alfabet database connections can then be configured for the same connection to the source database. For each Alfabet database connection, one or a subset of the object classes that can be imported via the connection can be selected as allowed object classes.

- In the **Import Data Search** functionality, the user first selects an Alfabet database connection. He/she then selects one of the object classes configured to be allowed object classes for the Alfabet database connection. Optionally, a full text search filter can be used to further restrict the import to a subset of

the selected object class. As a result, import is limited to a subset of objects of one object class at a time. This is also relevant for performance, because the transmission of object data is limited to the objects found by the filtering.

The configuration includes a number of access permission settings to ensure that import can be restricted to a defined user for a defined object class.

The following configuration steps are required to activate the functionality:

- Configuring the Source Database to Provide Access to the Relevant Data via the Alfabet RESTful Services

- Configuring the XML Object AlfabetIntegrationConfig of the Target Database

- Creating an Alfabet Database Connection in the Target Database

- Providing Access to the Import Data Search Functionality

- Optional Configuration for Detection of Changes to Configuration Object

Configuring the Source Database to Provide Access to the Relevant Data via the Alfabet RESTful Services

The source database must be configured to allow access to the endpoint `objects` and `metamodel` of the Alfabet RESTful services. The required configuration is described below in a short overview. For a detailed description of the required configuration of the see the Alfabet RESTful API reference manual *Alfabet RESTful API*.

- A license for the Alfabet Data Integration Framework (ADIF) must be active.

- Configure the `web.config` file of the Alfabet Web Application to enable Alfabet RESTful services.

- Configure the Alfabet Web Application on the Web Server to enable access to the Alfabet RESTful services.

- Configure the server alias of the Alfabet Web Application to enable access to the Alfabet RESTful services.

- Create a user for access via the Alfabet RESTful services, with **Has GetObjectsByFilters Access** and **Has Meta-Model Access** permissions to the Alfabet RESTful services and generate a user password for the user.

- For each object class that shall be imported, define a class setting with the attribute **Allow Read via Rest API** set to `true`.

- Assign the relevant class settings to a user profile via a view scheme and assign the user profile to your access user.

- If your company is using Alfabet 's mandate concept, make sure that the mandate settings for your access user allow viewing all relevant objects.

In addition to the configuration relevant for the Alfabet RESTful services, the object classes that shall be imported into the target database may be configured to have an object class property `LAST_UPDATE` that enables to detect whether a configuration object that exists in both the source and target object has been changed in the source database after the last change to the object in the target database. For more information about the complete configuration to activate this feature, see *Optional Configuration for Detection of Changes to Configuration Object*.

Configuring the XML Object AlfabetIntegrationConfig of the Target Database

The Alfabet Web Application of the target database acts as RESTful client, connecting to the RESTful API of the Alfabet Web Application of the source database. The connection parameters must be configured in the configuration of the target database using Alfabet Expand:

1) Open the **Presentation Model Designer** of Alfabet Expand and expand the explorer node **XML Objects** > **Integration Solutions**.

2) Click the XML object `AlfabetIntegrationConfig`. The attribute window opens.

3) Click in the attribute **XML Definition**. The XML object opens in a text editor.

4) Edit the XML as described below.

5) Click **OK** to close the editor.

6) Click the **Save** button to save your changes.

The XML object must contain the following XML structure:

```
<AlfabetIntegrationConfig>

    <Connection

        name = "My Test Import"

        service="http://localhost/ALFABET1"

        active="true"

        data_portion="80"

        search_limit="500"

        emptyValues="true"

        user="DAME"

        psw="H7GLVUGZWQETKJFSX7HY6OK2CUB4WRGK"

        profile="MASTER">

            <ImplementedClass parent ="IncomeType" />

            <ImplementedClass parent ="ObjectMonitor" />

    </Connection>

</AlfabetIntegrationConfig>
```

The following XML elements and their attributes are part of the specification:

Element (Bold) / Attribute	Allowed Values	Mandatory/Optional	Configuration Requirements
AlfabetIntegrationConfig		Mandatory	Root node of the configuration
Connection		Mandatory	The configuration parameters for sending the RESTful service call and request the data from the source database.

Element (Bold) / Attribute	Allowed Values	Mandatory/Optional	Configuration Requirements
			This XML element can be added multiple times to define multiple source databases.
name	String	Mandatory	Enter a unique name for the connection configuration. The name is used to identify the connection configuration for example in editors on the Alfabet user interface.
service	URL	Mandatory	Enter the URL of the Alfabet Web Application for access to the source database.
active	true/false	Optional, the default is true	This attribute can be defined to deactivate (false) or activate (true) the connection. If the attribute is set to false, the connection will not be displayed in the drop-down list for the connection specification in the editor **Alfabet Database Connection** on the Alfabet user interface. In the **Import Data Search** functionality, all Alfabet database connections already defined for the connection definition are not available for triggering data import.
data_portion	Integer	Optional, the default is 100.	Enter the maximum number of results to be returned in a single request. If the maximum number of search results exceeds the data_portion value, multiple requests are send to the Alfabet RESTful API of the Alfabet Web Application connected to the source database, each returning a subsequent subset of the data, until the maximum overall number of search results defined with the attribute search_limit is reached.
search_limit	Integer	Optional, the default is 300.	Enter the maximum number of objects for that the response of the call shall return data. This value may be set to limit data return from databases containing a high number of objects. **NOTE**: Import of a very high number of objects may lead to performance issues. It is recommended to restrict the maximum number of objects to be returned and use the filters on the **Import Data Search View** to limit the number of actually imported data to a subset of the data available in the source database.
emptyValues	true/false	Optional	If set to true, all object class properties for an object are included into the response call, even if the property value is NULL. If set to false, only object class properties that are set in the source database are transmitted. It is recommended to set the parameter to false for transmission of high amounts of data. In the target database, values for a property are set to NULL for an object if the response call

Element (Bold) / Attribute	Allowed Values	Mandatory/Optional	Configuration Requirements
			does not contain a value for a property or if it contains an empty field for the property.
user		Mandatory	Enter the user name of the user configured in the source database to grant access to the functionality. For more information about the configuration of the user, see .
psw		Mandatory	Enter the REST API password of the user configured in the source database to grant access to the functionality. For more information about the configuration of the user, see .
profile		Mandatory	Enter the user profile configured in the source database to grant the correct access permissions to the relevant object classes.
Implement- edClass		Mandatory	Defines which classes can be imported from the source database. Multiple XML elements `ImplementedClass` can be added to the XML object to define import from all relevant classes within one `Connection` definition.
parent		Mandatory	Enter the name of the object class to be imported or the name of the parent class in the class hierarchy of the Alfabet meta-model. Import will be allowed for the specified object class and all object classes that are subordinate of the specified object class in the Alfabet meta-model.

The definition of the object classes for import can be done via the parent object class to ease configuration. The hierarchy in the Alfabet meta-model is not visible in the Meta-Model tab in Alfabet Expand. If you have a valid license for ADIF, you can see the class hierarchy in the explorer of the ADIF tab in Alfabet Expand.

The following table lists all object classes that are relevant for import and the parent object class.

The table also informs whether the LAST_UPDATE property required for the optional data update functionality is available for the object class and whether it is available by default or requires configuration.

If the object class has standard object class properties of the type Reference or ReferenceArray, this information is also given in the list. Properties of the type ReferenceArray are ignored during import and properties of the type Reference are only imported if the target object for the reference is already available in the target database. If you import for example configured reports structured in report folders, you must allow import of both classes and report folders must be imported first.

If you define a parent object class, you might technically be able to import data for a high number of object classes. Nevertheless, object classes not listed here should not be imported via the **Import Data Search** functionality.

Class caption	Object class name	Parent object class name	Update functionality	Import restrictions
Currency	Currency	ITClass	No	Currencies are hierarchically structured. The currencies must be imported from top level first to the bottom level last.
Cost Type	CostType	ITClass	Requires configuration	Cost types are hierarchically structured. The cost types must be imported from top level first to the bottom level last.
Income Type	IncomeType	ITClass	Requires configuration	Income types are hierarchically structured. The income types must be imported from top level first to the bottom level last.
Connection Type	ConnectionType	ITClass	Requires configuration	
Connection Method	Connection-Method	ITClass	Requires configuration	
Connection Frequency	ConnectionFre-quency	ITClass	Requires configuration	
Connection Data Format	ConnectionData-Format	ITClass	Requires configuration	
Role Type	RoleType	ITClass	Requires configuration	
Skill	Skill	ITClass	Requires	

Class caption	Object class name	Parent object class name	Update functionality	Import restrictions
			configuration	
Data Retention Policy	DataRetention-Policy	ITClass	Requires configuration	
Indicator Time Series	Indica-torTimeSeries	ITClass	Requires configuration	
Generic Reference Data	GenericRefer-enceData	Artifact	Available by default	
Alfabet Database Connection	Alfabet_DBCon-nection	Integra-tionCon-nection	Available by default	Assignment of authorized user groups is based on a ReferenceArray property and will not be included into the import. For security reasons, the development and production environment should use a different user configuration and authorized access shall not be imported but configured in the target database.
API Gateway Database Connection	APIGate-way_DBConnec-tion	Integra-tionCon-nection	Available by default	Assignment of authorized user groups is based on a ReferenceArray property and will not be included into the import. For security reasons, the development and production environment should use a different user configuration and authorized access shall not be imported but configured in the target database.
API Portal Database Connection	API-Portal_DBCon-nection	Integra-tionCon-nection	Available by default	Assignment of authorized user groups is based on a ReferenceArray property and will not be included into the import. For security reasons, the development and production environment should use a different user configuration and authorized access shall not be imported but configured in the target database.
ARIS Database	ARIS_DBCon-nection	Integra-tionCon-nection	Available by default	Assignment of authorized user groups is based on a ReferenceArray property and will not be included into the import. For security reasons, the development and production environment should use a

Class caption	Object class name	Parent object class name	Update functionality	Import restrictions
Connection				different user configuration and authorized access shall not be imported but configured in the target database.
CentraSite Connection	CentraSite_DBConnection	IntegrationConnection	Available by default	Assignment of authorized user groups is based on a ReferenceArray property and will not be included into the import. For security reasons, the development and production environment should use a different user configuration and authorized access shall not be imported but configured in the target database.
Report Folder	ALFA_REPORTFOLDER	ALFA_REPORTBASE	Available by default	Assignment of authorized user groups is based on a ReferenceArray property and will not be included into the import. For security reasons, the development and production environment should use a different user configuration and authorized access shall not be imported but configured in the target database.
Configured Report	ALFA_REPORT	ALFA_REPORTBASE	Available by default	If configured reports are structured in report folders, the report folder must be imported prior to importing the configured reports. Access permission configuration will not be included into the import. For security reasons, the development and production environment should use a different user configuration and access permissions shall not be imported but configured in the target database.
Consistency Monitor	ConsistencyMonitor	ObjectMonitor	No	The listeners of the monitor are stored as ReferenceArray and are not included into the import. For security reasons, the development and production environment should use a different user configuration and listeners are then anyway not meaningful to import.
Notification Monitor	NotificationMonitor	ObjectMonitor	No	The listeners of the monitor are stored as ReferenceArray and are not included into the import. For security reasons, the development and production environment should use a different user configuration and listeners are then anyway not meaningful to import.
System Date Monitor	SystemDateMonitor	ObjectMonitor	No	The listeners of the monitor are stored as ReferenceArray and are not included into the import. For security reasons, the development and production environment should use a different user

Class caption	Object class name	Parent object class name	Update functionality	Import restrictions
				configuration and listeners are then anyway not meaningful to import.
Mandate	ALFA_MANDATE	SYSTEM-CLASS	No	

Creating an Alfabet Database Connection in the Target Database

Configuration is done in the **Integration Solutions Configuration** functionality on the Alfabet user interface.

1) Go to the **Integration Solutions Configuration** functionality and click the **Alfabet Database Connection** node in the **Integration Solutions Configuration** explorer.

2) In the view, click **New** > **Create Alfabet Database Connection**.

3) In the **Alfabet Database Connection** editor, define the following fields as needed:

Basic Data tab:

- **ID**: Alfabet assigns a unique identification number to each ARIS database connection. This number cannot be edited.

- **Name**: Enter a unique name for the Alfabet database connection. The name is displayed in the **Import Data Search** functionality in the drop-down list for selection of the master database for import of Alfabet configuration data.

- **Release Status**: Select the Alfabet database connection's current release status.

  > The set of release statuses available for an object class are configured by your solution designer in the configuration tool Alfabet Expand. For more information, see the section *Configuring Release Status Definitions for Object Classes* in the reference manual *Configuring Alfabet with Alfabet Expand*. For general information about release statuses, see the section *Understanding Release Statuses* in the reference manual *Getting Started with Alfabet*.

- **Description**: Enter a meaningful description that will clarify the purpose of the Alfabet database connection.

Authorized Access tab:

- **Authorized User**: Click the **Search** icon to assign an authorized user to the selected ARIS database connection. The authorized user will have Read/Write access permissions for the object and is responsible for the maintenance of the object.

- **Authorized User Groups**: Select one or more checkboxes to assign Read/Write access permissions to all users in the selected user group(s).

Connection tab:

- **Alfabet Connection**: Select the connection to the relevant Alfabet database that is configured in the XML element **Connection** in the XML object **AlfabetIntegrationConfig** in Alfabet Expand.

- **Allowed Classes**: Select one or multiple object classes for that users authorized to import data via this Alfabet database connection shall be able to import data via the **Import Data Search** functionality. All classes that may be imported according to the definition in the XML element **Connection** in the XML object **AlfabetIntegrationConfig** in Alfabet Expand are listed in the drop-down list of the field.

  After you have selected object classes for import, you will see your selection in the field **Selected Classes**. The user will see the selected classes in the drop-down list of the object class filter in the **Import Data Search** functionality after having selected the Alfabet database connection.

4) Click **Test Alfabet Database Connection**. If your settings are correct, a message "The connection is valid" is displayed. Otherwise an error message is displayed.

Providing Access to the Import Data Search Functionality

In the target database, the user that shall import the data via the **Import Data Search** functionality must log in with a user profile that is configured to grant access to the functionality per object class. The following must be configured for the user profile:

- The user must be able to access the **Import Data Search** functionality (`ImportDataSearch`) either via a menu item or a guide view or guide page.

  (i) For information about how to add a functionality to a user profile, see *Making Functionalities Accessible to a User Profile*.

- The attribute **Allow Update from External Reference Data Service** must be set to `true` for all class settings of all object classes for that objects shall be imported. The relevant class settings must be assigned via the view scheme configuration to the user profile the user logs in with.

  (i) For information about how to configure class settings for an object class, see *Configuring Class Settings for Object Classes and Object Class Stereotypes*.

  For information about how to assign class settings to the view scheme of a user profile, see *Configuring a View Scheme for a User Profile*.

Optional Configuration for Detection of Changes to Configuration Object

Optionally, the import mechanism can be configured to detect whether a configuration object that exists in both the source and target object has been changed in the source database after the last change to the object in the target database. This mechanism is exclusively based on the value of the `LAST_UPDATE` property of the object.

Some of the configuration objects already have a standard property called `LAST_UPDATE`. For these object classes the functionality is available by default. For other object classes, the functionality can be implemented by adding a custom property `LAST_UPDATE` to the object class in both the source and the target database. A custom property with the Name attribute set to `LAST_UPDATE` will automatically be set to the current data by the system whenever the object is changed.

If a `LAST_UPDATE` property is available for the object class in both source and target database the search functionality of **Import Data Search** automatically compares the dates stores in the custom or standard properties `LAST_UPDATE` of the object if the source and the object with the same image property values in the target database. No further configuration is required.

If the `LAST_UPDATE` date of the target database is older than the one in the source database, the object is marked as changed. If the object in the source database is changed on the same day as the object in the target database, the change cannot be detected, because the LAST_UPDATE property only safes the date and not the date and time of the update.

For a subset of object classes relevant for import a `LAST_UPDATE` property is neither available by default nor can it be added via configuration. A complete overview of the availability of the `LAST_UPDATE` property is given in the list of relevant object classes provided in the section *Configuring the XML Object AlfabetIntegrationConfig of the Target Database*.

To create a custom `LAST_UPDATE` property, you must perform the following action in Alfabet Expand:

1) Open the **Class Designer** and expand **Meta-Model** > **Classes** in the explorer.

2) In the explorer, click on the object class for that you want to define a `LAST_UPDATE` property.

3) Click on the arrow on the right and select **Add New Property**.

4) In the window that opens, enter `LAST_UPDATE` into both the **Property Name** and **Technical Name** field. `LAST_UPDATE` must be written in capital letters.

5) Click **OK**. The new property is added to the explorer tree and the attribute window shows the attributes of the new property.

6) In the **Property Type** attribute, select `Date` from the drop-down list.

7) Click the **Save** button to save your changes.

Importing Configuration Objects from a Source Database

The import of configuration relevant objects from a source to a target database can be performed under the following preconditions:

- The connection and data access permissions are fully configured as described in the section above.

- Both databases are currently available via a running Alfabet Web Application.

Import is done in the functionality **Import Data Search** and is a performed in two steps:

- First a comparison between the data in the source database and your current database is triggered. The result is displayed in a table.

- Import can then be performed by selecting one or multiple objects in the table and triggering import via a button in the toolbar.

⚠️ Prior to starting an import, please carefully consider the restrictions that apply to data import that are listed in the section *Importing Objects of Configuration Relevant Object Classes from a Master Database*!

Do the following to import objects from the source database in the functionality **Import Data Search**:

1) Set the following filters:

- **Master Connection**: Select the Alfabet database connection that connects to the source database. You will see all Alfabet database connections that have been configured by your solution designer in the functionality **Integration Solutions Configuration**.

- **Search for**: Select the object class for that you want to import objects. The drop-down list contains all object classes that are configured in the Alfabet database connection as allowed object classes and for that you have import permissions granted via the user profile you are logged in with.

- **in**: After selection of an object class in the **Search for** field, the drop-down list of the field **in** is automatically filled with all searchable properties of the class. These are all object class properties with the **Property Type** set to `String` or `Text`. You can deselect properties from the drop-down list to search in a sub-set of the searchable properties only. Please note that for classes that are not visible in the **Classes** explorer in the **Meta-Model** tab of Alfabet Expand, search is limited to finding objects with a defined name.

- **Search Pattern**: Enter the string to find in the object class properties selected in the **in** field. The search finds objects in both source and target database. An asterisk can be user as wildcard in the search pattern.

2) Click the **Search** button. Results are listed in the table. The table lists all objects matching your filter definition with their name, description and, if applicable, their short name. The column **Data Source** and the color coding for the rows in the table informs whether the objects are found in the target database or in the source database only:

Object available in current (tar-get) database	Object available in source database	Value for Data Source	Row color
yes	no	object is not included into the table	
yes	yes, `LAST_UPDATE` date identical or prior to `LAST_UPDATE` date of the object in target database or objects do not have a `LAST_UPDATE` property	Target	no back-ground color
yes	yes, `LAST_UPDATE` data younger than `LAST_UP-DATE` date of the object in the target database	Target to Update	blue
no	yes	Master	green

3) Select one or multiple new or changed objects in the table and click **Create as Copy** to copy the object into your database.

ℹ️ If the Alfabet user interface is rendered in a secondary language and data translation is enabled for the currently used interface language, the search results are displaying the translated object class properties, if provided. For objects for that no translation is provided, the string is displayed in the standard language (English, en-US). Please note the following for search and import of data:

- The search refers to the object class property values as displayed. That means if a translated value is displayed, the search finds the object by the translated value. If the English value is displayed, the user must enter the English string to find the object.

- The data import compares the values provided in the standard language, which is English (en-US). If for example objects are identified by name and the translation of the name changes while the name in the original language is not changed, the object is detected as changed (if this feature is implemented for the class) and not as a new object.

- If automated data translation capability is implemented, all automated translations for objects that have be imported are handled as automated translations in the target database.

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 the upload is performed will depend on how 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. The upload of assemblies to a database using the Alfabet Administrator is described in the section *Managing Assemblies* in the reference manual *System Administration*.

- 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 the section *Uploading Assemblies via the *.amm Update File to Another Database* in the reference manual *System Administration*.

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 functionality to take over the configuration of a master database to a target database can be used in the test or production environment to take over the DLL files and the required configuration changes that have been made in the development environment. If the execution of a script is required, this has to be performed separately by a system administrator.
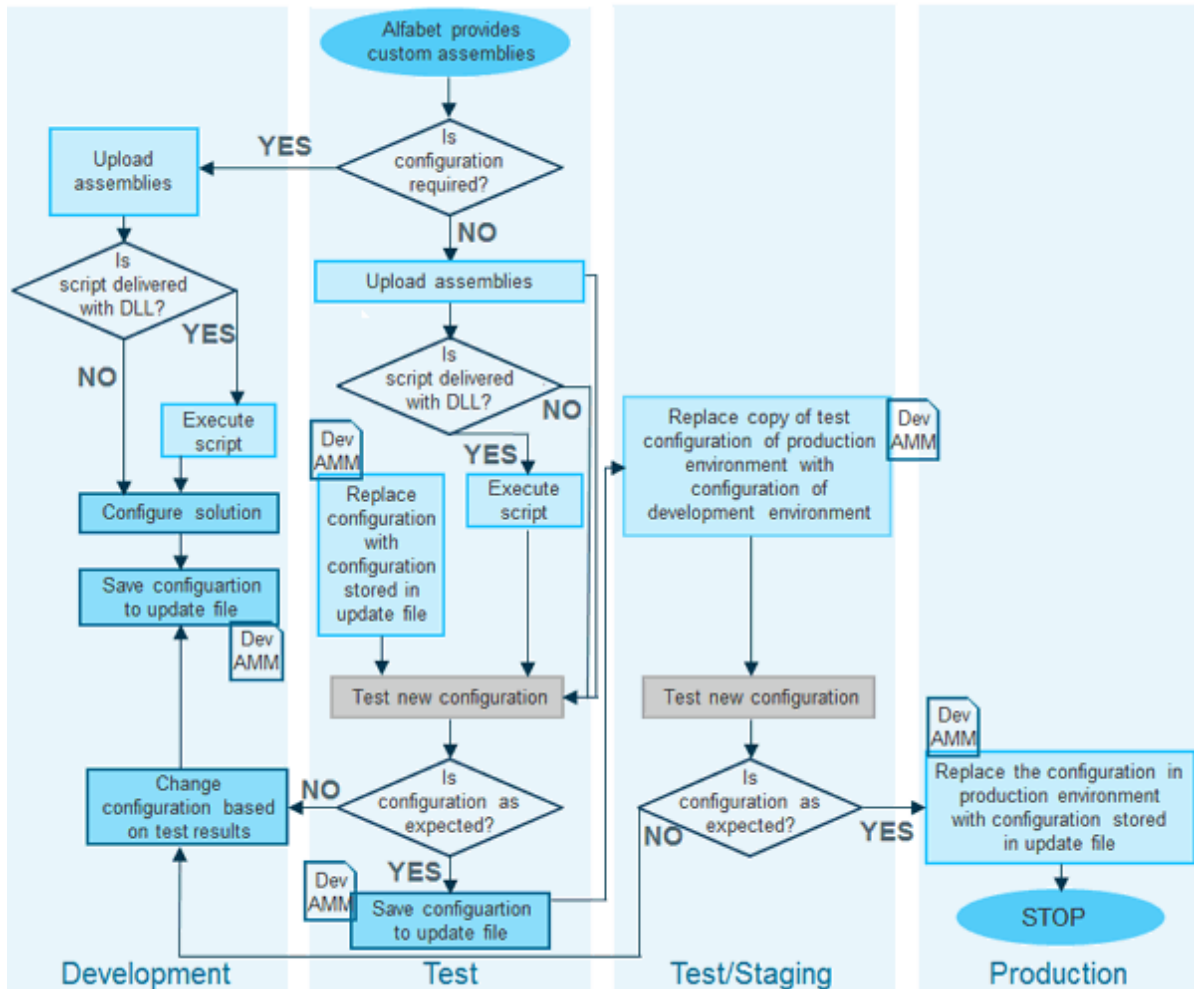
FIGURE: Example for a best practice workflow for the upload of custom assemblies to the database

Anonymizing Data

The data anonymization capability ensures data transparency and accountability across the enterprise as well as meet requirements of the General Data Protection Regulation (GDPR), and security requirements for business operations. For example, by means of pseudonymization, user data can be replaced with an artificial identifier to ensure anonymity if the user leaves the enterprise, or data about the architecture elements in IT can be used in the production environment but all sensitive data can be replaced with artificial identifiers in the development or test environment.

Anonymization can be performed on data of the type `String`, `Text`, `URL` and `Picture`. During anonymization, the original data can either be set to NULL, or be replaced with a random string or the REFSTR of the current object. Replacing the data with the REFSTR of the current object ensures that there is an unambiguous assignment of the data to the object. Therefore, the anonymization of key values, like the Name of an object, the User Name property for Alfabet users, or of object class properties defined as unique keys for a property is restricted to this method. Instead of the Name property of an object class, the solution administrator can decide during configuration of the feature to specify another property as key property that will then be restricted to being replaced with the REFSTR of the object during anonymization.

Anonymization requires configuration of individual object classes and properties thereof to be included into anonymization. Triggering anonymization via one of the available mechanisms then leads to anonymization of all data

of all object classes for that anonymization is configured. The data is anonymized in the database table for the object class as well as in the audit history table for the object class. If user data is anonymized and the User Name of the user is included in anonymization, the User Name is also anonymized in all audit history tables of all object classes and in the information about the creation user and last update user available for configuration objects in Alfabet Expand.

For user data an additional method for anonymization is available. Data can be anonymized for individual users only. In addition, individual users can be excluded for anonymization. If data is anonymized in general, these users are not anonymized, which will for example ensure that administrators are still able to log in to Alfabet with their user name while all other user names were changed during anonymization.

Anonymization is performed via the Alfabet Server the Alfabet Web Application is configured to connect to. Prior to performing anonymization, a solution designer must pre-configure the anonymization capability as described in the following.

The following information is available:

- Activating Anonymization for Object Class Properties

  - Changing Key Property Settings for Object Classes

  - Configuring the Object Class to be Anonymized

- Configuring Anonymization of Data of Single Users Only

- Excluding Users from Anonymization

- Anonymizing Data

  - Anonymizing All Relevant Data in the Alfabet database

  - Anonymizing Data of Selected Users

  - Creating a Database Archive File Containing Anonymized Data

- Checking Anonymization Actions

Activating Anonymization for Object Class Properties

The following steps are involved in the activation and configuration of anonymization:

- Changing Key Property Settings for Object Classes

- Configuring the Object Class to be Anonymized

Changing Key Property Settings for Object Classes

It is crucial to maintain data integrity during anonymization to be able to work with a database that contains anonymized data without any restrictions in usability.

A number of mechanisms are implemented that ensure data integrity:

- If an object class property of the type `String` is based on an enumeration, it is excluded from anonymization.

- The following object class properties are either based on values defined for example in an XML object or are crucial for implemented functionalities and are therefore excluded from anonymization:

  - The object class properties `Stereotype`, `ObjectState`, `State`, `Status` and `Status History` for all object classes for which they are available.

  - The object class properties `CheckInStatus` and `CheckInProtocol` of the object class `Project`.

  - The object class properties `LevelID` and `LevelIDNum` of the object classes `Business Process` and `Domain`.

- For the object class `PERSON`, the object class property values of single objects can be excluded from anonymization. A new boolean property `ExcludedFromAnonymization` has been added to the object class `PERSON`. A new option **Exclude From Anonymization** has been added to the **User** editor to set this property. If the checkbox is selected, the data for the user cannot be anonymized.

- If the object class property is mandatory, the anonymization method that sets the value to `NULL` is not available.

- If the object class property is included into the specification of a Class Key with a uniqueness constraint for the object class, the object class property can only be substituted with the REFSTR of the object during anonymization to ensure that the uniqueness constraint is observed.

- Independent from Class Key specifications for individual object classes, the `Name` property of an object class is by default regarded as key property and the object class property can only be substituted with the REFSTR of the object during anonymization. This rule can be overwritten by the solution designer in the XML object `AnonymizationKeyManager`. By default, the XML object `AnonymizationKeyManager` contains a definition for the object class `Person`. For the object class `Person`, the object class properties `USER_NAME` and `TECH_NAME` are set as key properties by default. This ensures that the audit history information and the information about the creator and last update user for configuration objects in Alfabet Expand, that is based on the `USER_NAME` or `TECH_NAME` property, is not corrupted.

  > ⓘ The server alias of the Alfabet Web Application can be configured with the attribute **Server Settings** > **General** > **Update History User Name** to use the `TECH_NAME` instead of the `USER_NAME` of the user for writing information about the `CREATION_USER` and `LAST_UPDATE_USER` into the audit history tables.

If you would like to change the key property settings in the XML object `AnonymizationKeyManager` in Alfabet Expand, you should do the following prior to activating anonymization for object class properties in the Alfabet Expand meta-model:

1) Open the **Presentation Model Designer** of Alfabet Expand and expand the explorer node **XML Objects**.

2) Click the XML object `AnonymizationKeyManager`. The attribute window opens.

3) Click in the attribute **XML Definition**. The XML object opens in a text editor.

4) For each object class for that you want one or multiple object class properties to be used as key properties instead of the Name property only, add an XML element `ClassAnonymizationKeyDef` to the root XML element `AnonymizationKeyManager`.

5) Set the following XML attributes for the new XML element `ClassAnonymizationKeyDef`:

   - `ClassName`: Write the name of the object class that the key shall be defined for into the attribute.

- *AnonymizationKeyProperties*: Write the name of the object class property that shall be the key property into the attribute. If multiple object class properties shall be defined as key, the object class property names must be written comma separated into the attribute.

6) Click **OK** to close the editor.

7) Click the **Save**button to save your changes.

The following specification changes the key for the object class Business Function to both the Name and Level ID instead of the Name only:

```
<AnonymizationKeyManager>

    <ClassAnonymizationKeyDef ClassName="BusinessFunction"
    AnonymizationKeyProperties="Name,LevelID"/>

</AnonymizationKeyManager>
```

Configuring the Object Class to be Anonymized

Data of an object class is only anonymized if the object class is configured to be anonymized. By default, anonymization is deactivated in the configuration of all object classes.

The activation settings that are required can only be performed on a subset of classes and object class properties:

- Both the object class and the object class property must be protected or public.

- The object class property must be of the type:

    - `String`, not based on an enumeration

    - `Text`

    - `URL`

    - `Picture`

- The property is not explicitly excluded from anonymization. Currently, the following object class properties cannot be anonymized:

    - The object class properties `Stereotype`, `ObjectState`, `State`, `Status` and `Status History` for all object classes for which they are available.

    - The object class properties `CheckInStatus` and `CheckInProtocol` of the object class `Project`.

    - The object class properties `LevelID` and `LevelIDNum` of the object classes `Business Process` and `Domain`.

Do the following to activate anonymization for an object class:

1) Open the Class Designer and expand the node **Meta-Model** > **Classes**.

2) Click the object class that you want to activate anonymization for.

3) In the attribute window, select the checkbox in the **Anonymization** attribute.

> ⓘ If the object class does not have any object class properties that can be anonymized, the attribute is deactivated.

4) Click into the attribute **Property Anonymization** to open the **Anonymization Rules for Class Properties** table. The table lists all object class properties of the current object class that might be anonymized. If the table does not open, the object class does not have any object class properties that can be anonymized.

5) In the column **Anonymization Type** of the table, select one of the following anonymization methods for each property to be anonymized:

- `ToBeLeftUnchanged`: The values for the object class property are not changed. Anonymization is not applied.

- `ToBeNullified`: The values for the object class property are set to `NULL`.

- `ToBeRandomized`: The values for the object class property are substituted with a random string. The length of the string is identical to the length of the original string. If auditing is activated, identical values are replaced with the same string in the audit tables. This method is only available for properties of the type `String` or `Text`.

  > ⓘ The following randomization mechanisms are implemented for strings and texts:
  >
  > - Properties of the type `String`: A random string is created using the Generate Random Password function of the .NET library with the length of the original string as input. Random strings do not contain special characters.
  >
  > - Properties of the type `Text`: A random text is generated from the following base text:
  >
  >   ```
  >   lorem ipsum dolor sit amet consectetur adipisicing elit
  >   sed do eiusmod tempor incididunt ut labore et dolore
  >   magna aliqua ut enim ad minim veniam quis nostrud
  >   exercitation ullamco laboris nisi ut aliquip ex ea
  >   commodo consequat duis aute irure dolor in
  >   reprehenderit in voluptate velit esse cillum dolore eu
  >   fugiat nulla pariatur excepteur sint occaecat cupidatat
  >   non proident sunt in culpa qui officia deserunt mollit
  >   anim id est laborum
  >   ```
  >
  >   This randomization mechanism is creating random texts from this base text using the variable integer parameters `minWords`, `maxWords`, `minSentences`, `maxSentences`, `numLines`, and the variable boolean parameter `randomSize`. These parameters are a function of the data in the database that shall be randomized
  >
  >   2.000 random texts are created during an anonymization run and subsequently used to replacing the original text from the class property. This allows performance characteristics to be retained after anonymization of data.

- `ToBeReplacedByKey`: The values for the object class property are substituted with the `REFSTR` value of the current object. This method keeps the information about which object the anonymized data belongs to. This might be useful for example in case of user data. This method is only available for properties of the type `String`.

The columns in the table provide information about the object class property that should be considered when choosing the anonymization method:

- **Property Name**: The name of the object class property to identify the data that can be anonymized.

- **Property Type**: The data type the object class property. The data type determines the availability of methods to be selected. Only strings can be replaced with a key and only string and text properties can be randomized.

- **Is Mandatory**: Some protected object class properties are mandatory. A value must be provided for these properties to ensure operability of all features implemented in Alfabet. If `True` is displayed in this column, the object class property cannot be set to `NULL` during anonymization.

- **Is Anonymization Key** : If `True` is displayed in this column, the object class property is either defined as key for the object class in the XML object `AnonymizationKeyManager` or there is no definition for the object class in the `AnonymizationKeyManager` and the object class property name is `Name`. This property can neither be set to NULL nor be randomized during anonymization.

- **In Unique Index** : If `True` is displayed in this column, the object class property is part of a unique class key definition of the object class. This property can neither be set to NULL nor be randomized during anonymization.

- **Has Validator** : If `True` is displayed, the values for this object class property are validated against a validator defined in the attribute **Validator** of the object class property. The availability of methods is not restricted if a validator is defined but depending on the selected anonymization methods the validation might fail after anonymization. You should check the object class property validator settings prior to selecting a validation method.

Configuring Anonymization of Data of Single Users Only

For the object class `PERSON`, anonymization can not only be performed on a per class but also on a per object basis. That means that next to anonymizing data for all users that are currently stored in the Alfabet database, you can also anonymize data for one or multiple selected users. anonymization is triggered via Alfabet Expand Windows or on the Alfabet user interface and executed via the Alfabet Server. Prior to triggering user data anonymization, anonymization must be configured for the object class `PERSON` as described in the section *Configuring the Object Class to be Anonymized*.

Excluding Users from Anonymization

The object class property values of single users can be excluded from anonymization. A new boolean property `ExcludedFromAnonymization` has been added to the object class `PERSON` for that purpose.

The property is set in the **Users Administration** functionality in the Alfabet user interface:

1) Navigate to the **Users Administration** functionality.

2) In the table, select the user that you want to exclude from anonymization.

3) In the toolbar, click **Edit** ✎. The **User** editor opens.

4) Select the checkbox **Exclude From Anonymization**.

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

Anonymizing Data

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.

> ℹ️ For information about the required configuration to enable anonymization, see *Activating Anonymization for Object Class Properties*.

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 **Class Designer** , 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, which means 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.

Anonymization can be triggered using other Alfabet components:

- All data anonymization methods are available via Alfabet Expand Windows.

- In Alfabet Expand Web, the method **Anonymize Data** is available in the context menu of **Utilities** > **Meta-Model Configuration**.

- In the Alfabet Administrator, all anonymization methods are available as options in the context menu of the server alias. For more information, see *Anonymizing Data* in the reference manual *System Administration*.

- In the **Users Administration** functionality on the Alfabet user interface, the user administrator can select one or multiple users in the table and select **Action** > **Anonymize User** in the toolbar to trigger anonymization for the selected user(s). Form ore information, see *Anonymizing User Data* in the reference manual *User and Solution Administration*.

- Anonymization can be triggered via a service call to the new endpoint `anonymizeuser` of the Alfabet RESTful service to anonymize data for users that are found via specification of the user's `REFSTR` in the REST API request. For more information, see *Anonymizing User Data For Selected Users* in the reference manual *Alfabet RESTful API*.

- The console application `AlfaAdministratorConsole.exe` can be used to anonymized all object class property values configured to be anonymized in an Alfabet database or data for one or multiple selected users. For more information, see *Anonymizing Data* in the reference manual *System Administration*.

The following information is available:

- [Anonymizing All Relevant Data in the Alfabet database](#)

- [Anonymizing Data of Selected Users](#)

- [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.

⚠️ 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!**

⚠️ Please note that the connection to the Alfabet database is closed during the anonymization process. Re-login of the current user is performed automatically with no need to fill in a login screen with the user name and password of the last login prior to anonymization. If the object class property `USER_NAME` is anonymized for the object class `PERSON`, a re-login to Alfabet Expand is not possible after anonymization. To avoid problems with re-login, the user performing anonymization can be excluded from anonymization. Alternatively, `ToBeReplacedByKey` may be used as anonymization method, which replaces the user name with the `REFSTR` of the respective user. Automatic re-login will then fail, but you can re-login via the login screen with the `REFSTR` as user name. The `REFSTR` can for example be written from a configured report prior to performing anonymization.

To trigger data anonymization in Alfabet Expand:

1) Open the **Utilities** designer and click on **Meta-Model Configuration** in the explorer.

2) Click the arrow on the right and select **Anonymize Data** from the menu. A new window opens.

3) Check the information in the Summary field to evaluate whether the current anonymization configuration in your Alfabet database is as expected.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 that the attribute **Anonymize** 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 attributes **Creator** and **Last Update User** in the **Tech Info** section of the configuration objects in Alfabet Expand. The attribute **Anonymize** 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 attributes **Creator** and **Last Update User** in the **Tech Info** section of the configuration objects in Alfabet Expand is not anonymized. This means that the `TECH_NAME` is anonymized, while the `USER_NAME` is left unchanged and the server alias configuration specifies that the `TECH_NAME` is written into 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, while the `TECH_NAME` is left unchanged and the server alias configuration specifies that the `TECH_NAME` is written into the audit history.

  - `False`: Audit history tables and the attributes **Creator** and **Last Update User** in the **Tech Info** section of the configuration objects in Alfabet Expand are not anonymized.

- **Class**: For each object class for that data will be anonymized, all object class properties that may be subject to anonymization are listed with information about the configured anonymization method.

4) Click **OK**.

Checking Anonymization Actions

Anonymization actions on the current database are logged. Information is written into a log file that can be specified in a dialog that opens for triggering the anonymization from Alfabet Expand Windows, the Alfabet Administrator or the Alfabet user interface. When selecting an already existing log file, the log information in the file is appended, although a message in the file selector states that the file is overwritten.

The console applications write information into the standard log files for Alfabet console applications if defined in the command line.

In addition, anonymization actions are listed in the view available in Alfabet Expand in the Utilities designer via the option **Show Update Meta-Model Configuration History**. The option is available in the menu that opens if you click the **Meta-Model Configuration** node and then click the arrow at the right of the selection bar.The table provides information about the kind of anonymization action performed, the status of the anonymization, the user performing the anonymization and the update time. In addition, the version of the software that was implemented when the anonymization action was performed is displayed.

Index