



ALFABET REFERENCE MANUAL

Alfabet Interface for RESTful Web Services

Documentation Version Alfabet 10.4.0

Copyright © 2013 - 2019 Software AG, Darmstadt, Germany and/or Software AG USA Inc., Reston, VA, USA, and/or its subsidiaries and or/its affiliates and/or their licensors.

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

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.
<i>Italics</i>	Used for emphasis, titles of chapters and manuals. this Example: see the <i>Administration</i> 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	6
Chapter 2: Required Licenses	8
Chapter 3: Activating the Alfabet RESTful API on Server Side	9
Enable the Alfabet RESTful API in the web.config Files of the Alfabet Web Application	9
Configure the Alfabet Web Application on the Web Server to Enable Access to the Alfabet Web Application	11
Disable the WebDAV module of the Internet Information Services	11
Setting the Required Authorization for the api Folder	12
Configuring the Server Alias of the Alfabet Web Application to Enable REST API Calls	13
Chapter 4: Authorization	14
Required Configuration on Server Side	15
Generating an API Password for a User for General Use of the Alfabet RESTful API	15
Enabling Reports, ADIF Schemes, and Workflow Templates to be Executed via RESTful Service Calls	17
Configuring per Object Class Permissions for Reading from or Writing to the Alfabet Database	18
Controlling Access via Mandates	21
Controlling Access per Object via Access Permission Concepts	22
Required Implementation on Client Side	23
Overview of Access Permissions Required for Each Endpoint	24
Chapter 5: Configuring Handling of Date, Time and Number Formats for API Calls	28
Configuring a New API Culture	28
Using an API Culture in a Service Call	29
Chapter 6: Service Calls and Return Values	31
Exporting Information about the Complete Alfabet Class Model Including Enumerations and Culture Settings	34
Exporting Information about All or Multiple Classes of the Alfabet Class Model	40
Exporting Information about All or Multiple Enumerations in the Alfabet Class Model	43
Exporting Information About Object Data Stored in the Alfabet Database	46
Exporting Data about Objects with Defined REFSTR Values	47
Exporting Data about Objects of a Defined Object Class Matching a Filter Definition	52
Exporting Information about Objects Found by a Configured Report	60
Deleting Objects from the Alfabet Database	70
Creating and Updating Object Data in the Alfabet Database	73
Creating a New Object in the Alfabet Database	76
Changing the Properties of an Existing Object in the Alfabet Database	80
Creating or Updating a Relation Between Objects in the Alfabet Database	84
Archiving Objects from the Alfabet Database	86
Regenerating the Password of an Alfabet User	89
Anonymizing User Data for Selected Users	92
Starting a Workflow via RESTful Service Call	94
Starting an ADIF Import via RESTful Service Call	95
Starting an ADIF Export via RESTful Service Call	98
Exporting Information about the Content of the Internal Document Selector	101

Downloading Documents from the Internal Document Selector	104
Uploading Documents to the Internal Document Selector	105
Checking Whether the Alfabet Components Are Running	108
Chapter 7: Accessing the Alfabet User Interface from the External Application	110
Chapter 8: Testing the Alfabet RESTful API	111
Testing the Alfabet RESTful API	111
Configurations Required to Use a Swagger Editor for Testing	111
Index	113

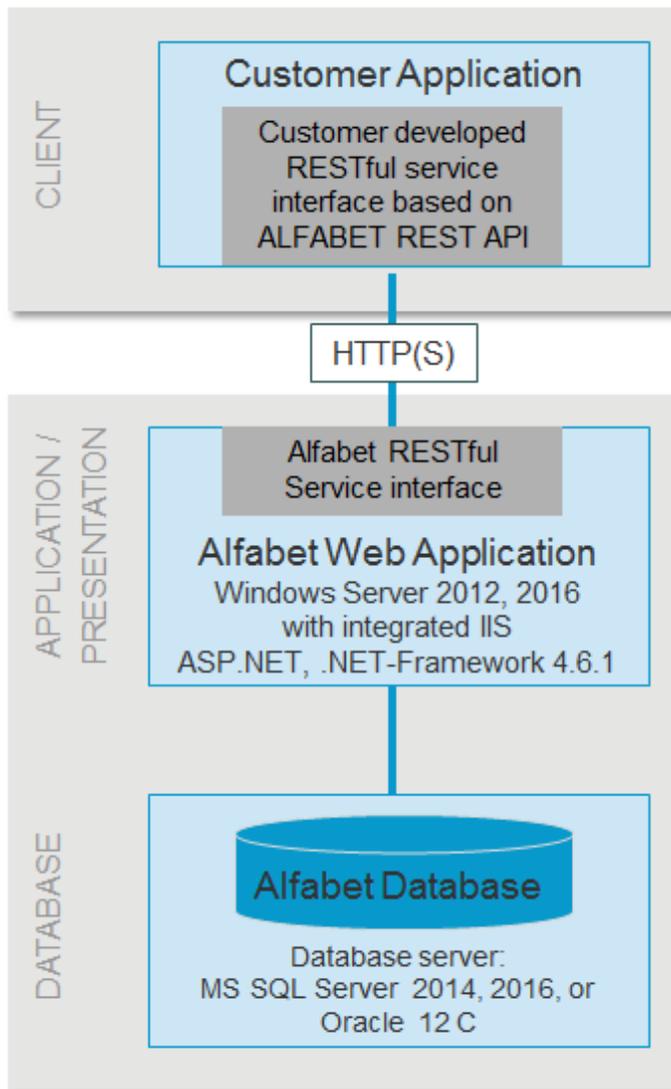
Chapter 1: Introduction

A RESTful API is available for the Alfabet application that provides easy access to the content in the Alfabet database. The API is designed as a web service architecture based on the Representational State Transfer (REST) software architecture type. Session cookies are not used.

The API can be used for the following:

- to get information about the structure of the object class model,
- to get information about objects stored in the Alfabet database,
- to create, update and delete objects and relationships in the Alfabet database,
- to trigger execution of ADIF jobs based on an ADIF scheme stored in the Alfabet database,
- to trigger start of a workflow via the Alfabet Web Application based on a workflow template stored in the Alfabet database.

This document describes the available data endpoints including the required calls and return values. On basis of the given information, customers can build interfaces that access the Alfabet database via HTTP request to the Alfabet RESTful API of a running Alfabet Web Application. The responses are sent in JSON format.



Chapter 2: Required Licenses

A license for the Alfabet Data Integration Framework (ADIF) must be active to use the Alfabet RESTful API.



In Alfabet 9.10, the Alfabet API for RESTful Web services has been extensively revised to enhance security. In Alfabet 10.4, both the previous version 1 of the Alfabet RESTful API as well as the new version 2 are supported, but this reference manual addresses only the newer version 2 of the Alfabet RESTful API. Please note that only the newer version 2 will be supported as of October 2017. Alfabet releases prior to 9.10 will not provide the functionality described in this reference manual.

The following GET request can be sent to the RESTful API of the Alfabet Web Application in order to check whether the Alfabet Web Application that is being connected to via RESTful services provides an interface based on version 1 or 2:

```
http://<URL of the Alfabet Web application>/api/Versions
```

The return value lists the available versions of the API on the server-side. This call enables the solution developer using the REST API for Alfabet to determine the functionalities that a specific Alfabet Server is to support. Please note that a version of the API must be enabled in the server alias of the Alfabet Web Application to be listed in the return value.

Chapter 3: Activating the Alfabet RESTful API on Server Side

Service calls to the Alfabet RESTful API of the Alfabet Web Application are only processed by the Alfabet Web Application if an active license for the Alfabet Data Integration Framework (ADIF) is available and the following configurations are done on server side:

- [Enable the Alfabet RESTful API in the web.config Files of the Alfabet Web Application](#)
- [Configure the Alfabet Web Application on the Web Server to Enable Access to the Alfabet Web Application](#)
 - [Disable the WebDAV module of the Internet Information Services](#)
 - [Setting the Required Authorization for the api folder](#)
- [Configuring the Server Alias of the Alfabet Web Application to Enable REST API calls](#)

Enable the Alfabet RESTful API in the web.config Files of the Alfabet Web Application

The `web.config` configuration file is located in the Alfabet Web Application directory of the installation directory for the Alfabet components. The `web.config` file can be modified using a standard text editor.

The required settings for the Alfabet RESTful API must be added to the `handlers` element of the `web.config` file of the Alfabet Web Application. The `handlers` element must have the following child elements apart from child elements that are already included for other processes:

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

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

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

In addition, the following code must be available in the `web.config` file to include the correct version of the required third party component `Newtonsoft.Json`:

```
<configuration>
...
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <dependentAssembly>
      <assemblyIdentity name="Newtonsoft.Json"
publicKeyToken="30ad4fe6b2a6aeed" culture="neutral" />
      <bindingRedirect oldVersion="0.0.0.0-12.0.0.0" newVersion="12.0.0.0" />
    </dependentAssembly>
  </assemblyBinding>
...
</runtime>
</configuration>
```

```
</runtime>
</configuration>
```



This section might be subject to changes during upgrades from one Alfabet version to the other, because of changes to the version of the embedded component. It is recommended to use the example `web.config` file delivered with the release and adapt it to the current environment to make sure that the above mentioned code and all other required settings are correctly set in the `web.config` file as required for the specific release.



If SAML or other single sign on authentication mechanisms are used for Alfabet, This settings might interfere with the settings required for the authentication method. Please contact Software AG Support if problems occur with the REST API settings in the `web.config` file.

Optionally, security of data transmission can be enhanced by specification of the JSON Web Token (JWT) for sending JSON objects via the Alfabet RESTful service API. By default, the JWT is hard-coded and therefore the same for all Alfabet installations. To change the JWT for an installation, an individual JWT must be base64 encrypted and the encrypted version must be entered into the in the `alfabet.config` file of the Alfabet Web Application:

- 1) Open the Alfabet Administrator.
- 2) Click the **Alfabet Aliases** node in the explorer. A workspace with a toolbar opens.
- 3) In the toolbar, click **Tools > Configure alfabet.config**. An editor opens.
- 4) Click the **Browse**  button on the right of the **Web Folder** field and select the main directory of the Alfabet Web Application from the directory browser. The `alfabet.config` file in the subdirectory `config` of the selected directory opens in the editor.
- 5) Add the following code as child element of the `alfaSection` XML element:

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

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

If your RESTful client will send a high number of service calls per second to the Alfabet RESTful service API, it might be required to increase the maximum allowed number of requests per second. By default processing of incoming RESTful service calls is limited to 300 per second. The limit can be changed in the `alfabet.config` file of the Alfabet Web Application:

- 1) Open the Alfabet Administrator.
- 2) Click the **Alfabet Aliases** node in the explorer. A workspace with a toolbar opens.
- 3) In the toolbar, click **Tools > Configure alfabet.config**. An editor opens.
- 4) Click the **Browse**  button on the right of the **Web Folder** field and select the main directory of the Alfabet Web Application from the directory browser. The `alfabet.config` file in the subdirectory `config` of the selected directory opens in the editor.
- 5) Find the `add` XML element with the `key` XML attribute set to `max_api_requests_per_second` and change the `value` XML attribute to the required number of requests per second. The default is 300:

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

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



Please note that request limits configured for the web server outside of the Alfabet Web Application for the maximum size of a request, the maximum URL length and the maximum length of a query string will apply to all calls to the Alfabet RESTful services.

Configure the Alfabet Web Application on the Web Server to Enable Access to the Alfabet Web Application

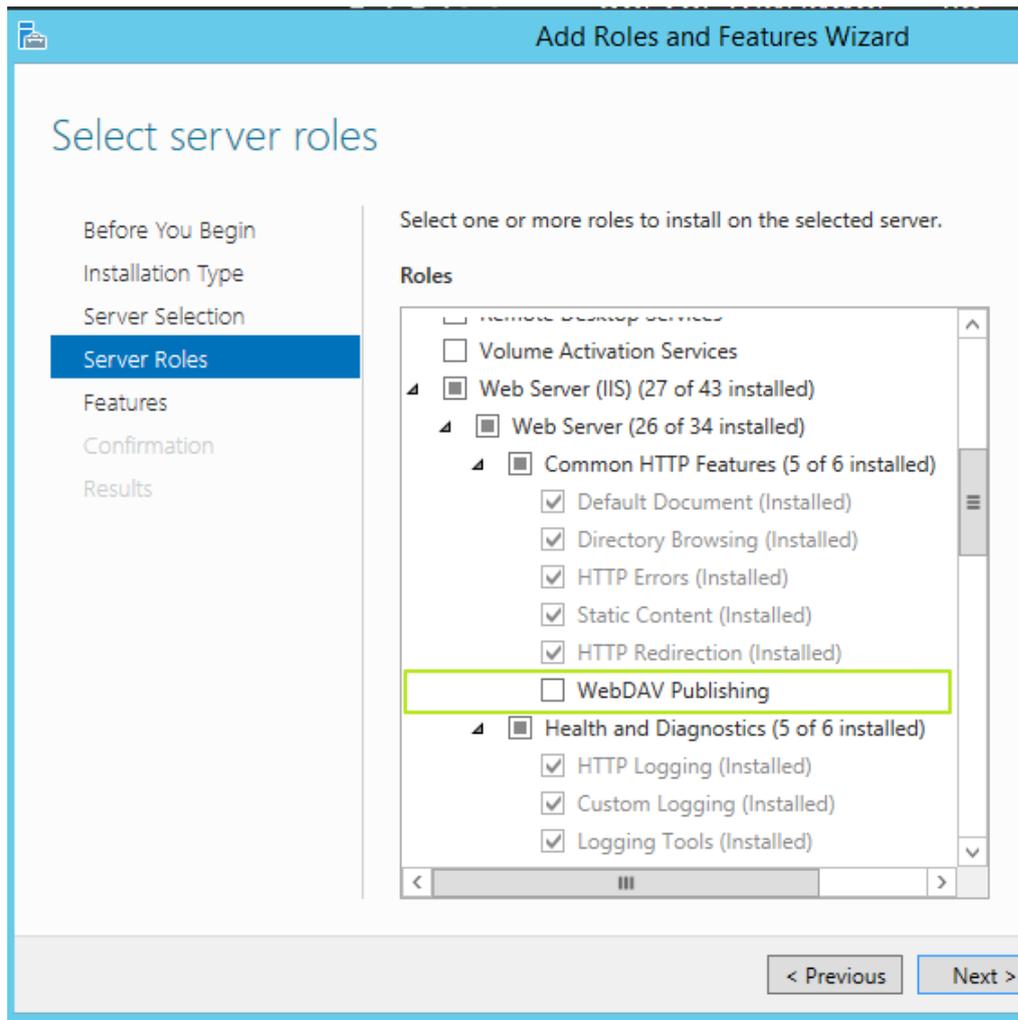
The following settings are required for the Web Server hosting the Alfabet Web Application to enable the Alfabet RESTful API:

- The WebDAV module of the Internet Information Services® hosting the Alfabet Web Application is not installed.
- The Alfabet RESTful API of the Alfabet Web Application does not support Windows sign On. Windows sign on must be disabled and Anonymous Access must be enabled for the `api` sub-folder of the Alfabet Web Application on the Web server.

Disable the WebDAV module of the Internet Information Services

The WebDAV module can be deactivated in the Server Roles:

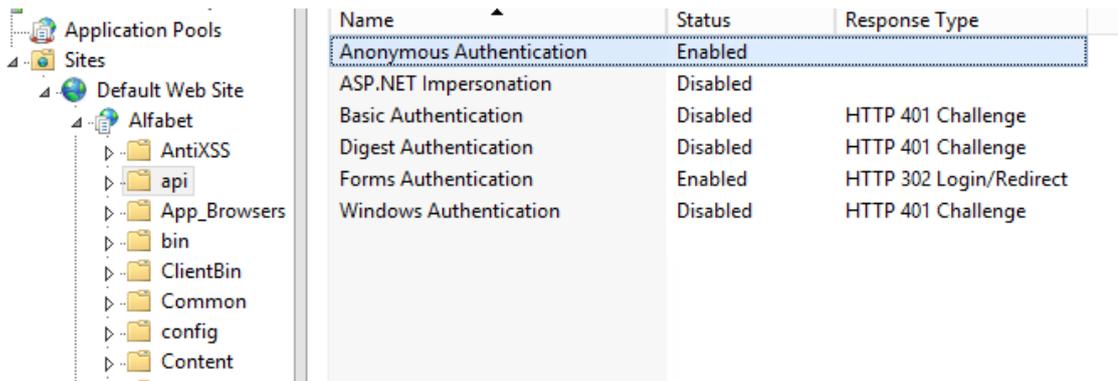
- 1) On the Web Server host, click the **Start**  icon that appears when you move the mouse to the lower left corner and click on the **Server Manager**  icon to open the Server Manager.
- 2) In the menu on the upper right of the Server Manager, select **Manage > Add Roles and Features**.
- 3) In the **Add Roles and Features Wizard**, select the installation type and the Web server host in the first three pages of the wizard and proceed to the **Server Roles** page, using the button **Next** of the wizard.
- 4) In the **Server Roles** page, expand **Web Server (IIS)**.
- 5) Make sure that the option **WebDAV Publishing** is not checked:



Setting the Required Authorization for the api Folder

Independent from the authorization selected for the Alfabet Web Application, the authorization of the `api` sub-folder of the Alfabet Web Application must be set to Anonymous Access:

- 1) On the Web Server host, click the **Start**  icon that appears when you move the mouse to the lower left corner and click on the **Server Manager**  icon to open the Server Manager.
- 2) In the menu on the upper right of the Server Manager, select **Tools > Internet Information Services (IIS) Manager**.
- 3) In the explorer, expand the node of the Alfabet Web Application and click on the `api` folder node.
- 4) In the section **IIS** in the middle pane, double-click **Authentication**. The status of the available authentication modes is displayed.



- 5) Set **Anonymous Authentication** to **Enabled** and **Windows Authentication** to **Disabled**. Changing of the settings is performed by selecting the option in the list and clicking the respective option **Disable** or **Enable** in the **Action** pane on the right.

Configuring the Server Alias of the Alfabet Web Application to Enable REST API Calls

The Alfabet API for RESTful Web services must be enabled in the server alias of the Alfabet Web Application. Configuration is performed in the configuration tool Alfabet Administrator:

 For information about how to access and use the Alfabet Administrator, see the section *Working With the Alfabet Administrator* in the reference manual *System Administration*.

- 1) In the explorer of the Alfabet Administrator, click the node **Alfabet Aliases**.
- 2) In the table, click the server alias that you want to configure.
- 3) In the toolbar, click the **Edit**  button. An editor opens.
- 4) Go to the **Server Settings > Security** tab.
- 5) Set the checkmark for the option **Enable REST API v2**.

 There is a second option **Enable REST API v1**. This option enables the use of the ARIS - Alfabet Interoperability Interface and is not relevant for common use of the Alfabet API for RESTful Web services.

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

Chapter 4: Authorization

Authorization for the Alfabet Data Integration Framework is done per user, per class and per object:

- Access to the Alfabet RESTful API must be explicitly granted to a named Alfabet user in the user configuration in the Alfabet database. The permissions can be restricted to a subset of the available endpoints.

Prior to calls for data request, an authorization call must be performed. In the authorization call the user name and token is submitted to the Alfabet RESTful API of the Alfabet Web Application, and a unique authorization key for data request calls is sent back. This unique authorization key can be used for approximately 20 minutes for data request calls following the authorization call. After the key expires, a new authorization call must be performed.

Users that are authorized to access the Alfabet RESTful API with the appropriate endpoint permission can read information about the structure of the Alfabet class model including enumerations and culture settings without further restrictions. Access to object data stored in the Alfabet database and to ADIF schemes, workflow templates and reports executed via REST API calls require additional permissions described below.

- Configuration objects like ADIF schemes, workflow templates, and configured reports are only executable via REST API calls if the attribute settings of the configuration object includes permission for execution via REST API. Event templates do not require a specific setting. They are exclusively designed to be used for REST API calls.
- Data about an object class can only be read, created or changed via the Alfabet RESTful API if the class settings for the object class are configured to grant the permissions. An object class can have multiple class settings. Class settings are assigned to view schemes. A view scheme is then assigned to a user profile. One or multiple user profiles can be assigned to a named user.

In parallel to login to the Alfabet user interface, REST API calls are also performed for a specific user with a specific user profile. The user profile is either defined in the REST API request or a default is selected from the user profiles assigned to the user. Whether the user is allowed to read, create or edit data for an object class, depends on the class settings relevant for the user profile.

- If the mandate capability is implemented in your company, the user can only read data about an object if the mandate settings for the user and the object permit access to the object. If the user has multiple mandates assigned, the mandate valid for the REST API call can be specified in the call. If the mandate specified in the call is not assigned to the user, an error is thrown.



For an overview of the mandate capability and the configuration of mandates, see *Implementing the Mandate Capability for a Federated Architecture* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- Whether a user can edit data about an object, depends on the implemented access permission concepts in your Alfabet solution, like for example right rules, the membership to authorized user groups, or the assignment of tasks about an object in the contexts of workflows or assignments.



For a complete overview of the object permission concepts in Alfabet, see the chapter *Configuring Access Permissions for Alfabet* in the reference manual *Configuring Alfabet with Alfabet Expand*.

Required Configuration on Server Side

Prior to sending service calls to the Alfabet RESTful API, the Alfabet solution must be configured to grant all required access permissions:

- [Generating an API Password for a User for General Use of the Alfabet RESTful API](#)
- [Enabling Reports, ADIF Schemes, and Workflow Templates to be Executed via RESTful Service Calls](#)
- [Configuring Per Object Class Permissions for Reading From or Writing Into the Alfabet database](#)
- [Controlling Access Via Mandates](#)
- [Controlling Access Per Object via Access Permission Concepts](#)

Generating an API Password for a User for General Use of the Alfabet RESTful API

Authorization is done per user and requires that access to the Alfabet RESTful API is explicitly granted to a named Alfabet user in the user configuration in the Alfabet database. The editor fields required for configuring the authorization are only available in the user editor if a valid license for ADIF is available.

User configuration is performed in the **User Administration** functionality in the user interface that is accessible via the `Admin` user profile or the **User Management** functionality that is available via the connected server alias node in the Alfabet Administrator.



For more information about how to create a named Alfabet user, see the chapter *Defining and Managing Users* in the reference manual *User and Solution Administration* or *Managing New and Existing Users* in the reference manual *System Administration*.

For security reasons and to avoid technical conflicts, take the following into account when granting REST API access permissions to a named user:

- The user should be exclusively used for calls to the Alfabet RESTful API and should not be used in parallel to access the Alfabet application via the Alfabet user interface or to access the Alfabet database via other interfaces provided by Alfabet, like ADIF or the ARIS - Alfabet Interoperability Interface.
- Multiple users with access to a subset to the endpoints of the Alfabet RESTful API that can either have read only or read/write access permissions can be defined. If multiple client applications are configured to send requests to the Alfabet RESTful API, individual users with access permissions exactly matching the requirements should be used for each client application.

After having created a named user, the following settings must be performed to grant access to the Alfabet RESTful API:

- 1) In the table of the **User Administration** functionality, select the user that shall be used to send request via the Alfabet RESTful API.
- 2) In the toolbar, click the **Edit**  button. The **User** editor opens.
- 3) In the editor, go to the **Permissions** tab.
- 4) Select the checkbox **Has API V2 Access**.
- 5) Specify details about the access permissions with the following attributes:

- **API Token Duration (minutes):** The RESTful service interface on client side must be implemented to send a request for authorization code prior to sending a data request. The authorization code received in the response of the authorization request can be used in data request calls to the Alfabet RESTful API posted within a limited amount of time, by default 20 minutes, after receiving the authorization code. The time period of the validity of the authorization code can be changed with this attribute. Enter the number of minutes the authorization code should be valid.
- **API Access Options:** By default, most of the options are checked in this field to give the user access to all central functionality provided for the Alfabet RESTful services. Deselect all options that the user should not have permissions to perform:
 - **Has Meta Model Access:** If selected, the user has access to the REST endpoints `metamodel`, `classes` and `enums` to read information about the structure of the Alfabet class model including enumerations and culture settings.
 - **Has GetObjectsByRefs Access:** If selected, the user has access to the REST endpoint `objects` to read information about data stored for objects in the Alfabet database that are found via specification of the object's `REFSTR` in the REST API request. The **Has GetObjectsByRefs Access** permission is also required for access to the `archiveobject` endpoint to export information about the object in an archive ZIP file containing relevant views as HTML files. Please note that access to object data is also controlled by per object class and per object permissions.
 - **Has GetObjectsByReport Access:** If selected, the user has access to the REST endpoint `objects` to read information about data stored for objects in the Alfabet database that are found via a configured report. Please note that access to object data is also controlled by mandate settings. In addition, access permission to the configured report must be granted via the user profile configuration and the attribute settings of the configured reports.
 - **Has GetObjectsByFilter Access:** If selected, the user has access to the REST endpoint `objects` to read information about data stored for objects in the Alfabet database that are found via specification of the object class and filter conditions in the call. Please note that access to object data is also controlled by per object class and per object permissions.
 - **Has CreateObjects Access:** If selected, the user has access to the REST endpoint `update` to create new object in the Alfabet database. Please note that object class specific permissions must also be granted.
 - **Has UpdateObjects Access:** If selected, the user has access to the REST endpoint `update` to update data for existing objects in the Alfabet database. Please note that object class specific and object specific permissions must also be granted.

In addition, this permission is required for access to the REST endpoint `regenerate-password`. Please note that the **Has PasswordAPI Access** permission must also be selected to use the `regeneratepassword` endpoint. Access to the user is also controlled by per object class permissions.

 - **Has DeleteObjects Access:** If selected, the user has access to the REST endpoint `delete` to delete existing objects in the Alfabet database. Please note that object class specific and object specific permissions must also be granted. The **Has GetObjectsByRefs Access** permission is also required for access to the `archiveobject` endpoint to export information about the object in an archive ZIP file containing relevant views as HTML files if the option for deleting the object after export of the archive is selected. Please note that access to object data is also controlled by per object class and per object permissions.
 - **Has AnonymizeUser Access:** If selected, the user has access to the REST endpoint `anonymizeuser` to anonymize users that are found via specification of the user's `REFSTR` in the REST API request in the Alfabet database.

- **Has PasswordAPI Access:** If selected, the user has access to the REST endpoint `regeneratepassword`. Please note that the **Has UpdateObjects Access** permission must also be selected to use the `regeneratepassword` endpoint. Access to the user is also controlled by per object class permissions.
- **Has ADIFAPIInvocationAccess:** If selected, the user has access to the REST endpoints `adifimport` and `adifexport` to trigger execution of ADIF jobs based on an ADIF scheme in the Alfabet database.
- **Has WorkflowAPIInvocation Access:** If selected, the user has access to the REST endpoint `workflow` to trigger start of a workflow based on a workflow template in the Alfabet database.
- **Has MonitoringAPI Access:** If selected, the user has access to the REST endpoint `monitor` to check whether the Alfabet components are available.
- **Has IDoc Upload Access:** If selected, the user has access to the REST endpoint `idocupload` for upload of documents into the internal document selector in the Alfabet database.
- **Has IDoc Download Access:** If selected, the user has access to the REST endpoint `idocdownload` for download of documents from the internal document selector in the Alfabet database.
- **Has IDoc File List Access:** If selected, the user has access to the REST endpoint `idocfilelist` for export of information about the content of the internal document selector in the Alfabet database.



For more information about granting per class permissions, see [Configuring Per Object Class Permissions for Reading From or Writing Into the Alfabet database](#). For more information about per object permissions, see [Controlling Access Per Object via Access Permission Concepts](#). An overview of all settings required for individual endpoints is given in the section [Overview of Access Permissions Required for Each Endpoint](#).

- 6) Click **Generate API Password**. A code is generated and stored in the Alfabet database.
- 7) Copy the code and store the information about user name and code for use on client side.

The generated code must be used for authorization of the client in requests sent to the Alfabet RESTful API of the Alfabet Web Application in combination with the user name.



The generation of the password can be performed for a user even if the checkbox **Has API Access** is not checked, but the generated code is only valid for access to the Alfabet RESTful API when the checkbox **Has API V2 Access** is set for the user.

Enabling Reports, ADIF Schemes, and Workflow Templates to be Executed via RESTful Service Calls

Some endpoints include execution of user configuration objects created by a solution administrator in Alfabet Expand:

- Configured reports can be executed via a service call to the endpoint `objects` to export predefined object data.
- ADIF Schemes can be executed via a service call to the endpoints `adifimport` or `adifexport`.
- Workflows can be started for a workflow template via a service call to the endpoint `workflow`.

The following configuration requirements must be met to execute the configuration objects:

- **For workflow templates:** Only workflow templates with the attribute **Automatic Start** set to `True` can be executed via RESTful service calls.
- **For ADIF schemes:** ADIF schemes that shall be executed via a RESTful service calls must not involve files located on the local network.
- **For configured reports:** The configured report must return a tabular dataset.

In order to execute configured reports, ADIF schemes and workflow templates via REST API, the required permissions must be defined in the attributes of the configured report, ADIF scheme or workflow template in Alfabet Expand:

- 1) Click the configured report, workflow template or ADIF scheme that you want to execute via REST API in the respective explorer of Alfabet Expand.
- 2) Set the attribute **Applicable for REST API** of the configuration object to `True`.
- 3) Check whether the following additional configuration requirements are met:
- 4) In the toolbar, click the **Save**  button to save your changes.

Configuring per Object Class Permissions for Reading from or Writing to the Alfabet Database

If you would like object data to be read or written via the Alfabet RESTful API, you must define a user profile that grants the required permissions via the class settings assigned to the user profile's view scheme.

In Alfabet, the user profile a user is logged in with and the view scheme and class settings assigned to this user profile determine the scope of functionality available to the user.



For detailed information about the configuration of class settings, view schemes and user profiles see the chapter *Configuring User Profiles for the User Community* in the reference manual *Configuring Alfabet with Alfabet Expand*.

Please note that while the predefined user profile `Admin` provides access to all objects via the Alfabet user interface independent of access permission concepts, this user profile does not grant access to any object via the RESTful API.

Most of the configuration for a user profile and the underlying view scheme and class settings is irrelevant for the use of the Alfabet RESTful API. The setting required to grant permission to create, edit or delete objects for an object class is exclusively specified via two attributes in the class setting of the object class. It is recommended to define class settings for exclusive use via the Alfabet RESTful API and assign them only to a view scheme and a user profile explicitly defined for use via the Alfabet RESTful API. The user profile can then be assigned to users authorized to read or alter data via the Alfabet RESTful API only.

Configuration of class settings, view schemes and user profiles is done in Alfabet Expand. The following configuration is required:

- 1) Go to the **Presentation** tab and expand the **Class Settings** node.
- 2) In the **Class Settings** folder, navigate to the object class folder that you want to define class settings for, and right-click the public or private class setting template  that you want to copy

and select **New Class Settings as Copy** to create a new class setting for the selected object class.



The option **New Class Settings for Stereotype as Copy** must not be used in this context. The REST API permissions set for object class stereotypes are ignored. Permissions can only be defined on the level of the object class.

- 3) A copy of the class settings template  is added to the class setting folder. All attributes specified for the class setting that you copied will also be copied to the new class setting.
- 4) Click the new class setting template to open the attribute window and enter a technical name for the class setting in the **Name** attribute.



It is recommended that the name of the class setting indicates the view scheme that it is being defined for and the name of the view scheme indicates the user profile it is assigned to. In other words, the name of the class setting should thus indicate the user profile that it is associated with.

A validation mechanism checks for correct syntax when defining a technical name. For a list of the special characters that are not allowed, see the section *Defining Attributes for Configuration Objects* in the chapter *Getting Started with Alfabet Expand* in the reference manual *Configuring Alfabet with Alfabet Expand*.

- 5) Set one or both of the following attributes to `True`, depending on the rights that should be granted for the class:
 - **Allow Read via Rest API:** Set to `True` to allow reading data of objects of this object class via the Alfabet RESTful API endpoint `objects`.
 - **Allow Write via Rest API:** Set to `True` to allow update data of objects of this object class, creation of objects and deletion of objects via the Alfabet RESTful API endpoints `delete` and `update`.
- 6) In the toolbar, click the **Save**  button to save your changes.
- 7) Repeat steps 2) - 6) for all object classes that shall be editable via the Alfabet RESTful API.
- 8) Go to the **Presentation** tab and expand the **View Schemes** node.
- 9) To create the view scheme, do one of the following:
 - Right-click the **View Schemes** folder and select **New View Scheme** to create a new view scheme from scratch, or
 - Click a view scheme you want to copy and select **New View Scheme as Copy** to create a new view scheme based on an existing view scheme.
- 10) You will see the new view scheme  added to the **View Schemes** folder. In the attribute window, enter a name in the **Name** attribute.



A validation mechanism checks for correct syntax when defining a technical name. Furthermore, if the technical name of a configuration object is changed, the name will be correctly updated in other configuration objects referencing the changed object during design time. Please note that the name of a changed configuration object will not be updated in guide pages nor if the guide pages reference a configuration object listed in the **Show Usage** functionality. If you plan to change the name of a configuration object, the reference in the guide page should be changed prior to changing the configuration object name.

Furthermore, please note that names may not begin with an empty space nor include special characters. For a list of the special characters that are not allowed, see the section

Defining Attributes for Configuration Objects in the chapter *Getting Started with Alfabet Expand*.

- 11) You should now systematically determine whether each object class/object class stereotype requires a custom class setting or whether the standard class setting suffices for your needs. To open the **View Scheme** editor for the selected view scheme, double-click the view scheme you are configuring. The **View Scheme** editor opens in the center pane.
- 12) The table labelled **Object Class Configuration** displays object classes/object class stereotypes on the first level. Click the + to expand the table below the object class or object class stereotype that you want to configure. When you expand the object class node, you will see all existing standard class settings and custom class settings configured by your enterprise for the selected object class. To specify which class setting should be implemented in the selected view scheme, click in the **Use in View Scheme** column, set an X for the class setting to be implemented for the selected view scheme. Carry out this procedure for all object classes with a class setting configuration for the Alfabet RESTful API.
- 13) In the toolbar, click the **Save**  button to save your changes.
- 14) Go to the **Admin** tab, right-click the **User Profiles** node and select **New User Profile**. A new user profile  is displayed in the explorer.
- 15) In the attribute window, define the following attributes:
 - **Name:** Enter a caption for the user profile. This is the name of the user profile that users will see when logging in to Alfabet.
 - **Type:** If the user profile should have editing permissions, select `Read/Write`. If the user profile should only have viewing permissions, select `ReadOnly`.
 - **View Scheme:** Click the **Drop-Down**  button to select the relevant view scheme that applies to the user profile when accessed by an external application or via a hyperlink in an e-mail notification.
- 16) In the toolbar, click the **Save**  button to save your changes.

After having defined the user profile, you must assign the user profile to a user that is already defined to have access to the Alfabet RESTful API as described in the section [Generating an API Password for a User for General Use of the Alfabet RESTful API](#).

User configuration is performed in the **User Administration** functionality in the user interface that is accessible via the `Admin` user profile.

- 1) In the table of the **User Administration** functionality, select the user that shall be used to send requests to alter data via the Alfabet RESTful API.
- 2) In the toolbar, click the **Navigate**  button. The object profile for the user opens.
- 3) Click **Assigned User Profiles** in the **User's Solution Configuration** section. The **Assigned User Profiles** page view opens.
- 4) In the toolbar, click **New > Assign User Profiles**. The **User Profile** selector opens.
- 5) Select the user profile that you have configured for changing data via the Alfabet RESTful API.
- 6) Click **OK** to save your selection.

In each call to the endpoints returning or changing object data, the user profile must be specified with the field `CurrentProfile` in the JSON payload of the request. If this is applicable, the information is given in the description of the service call in the corresponding section of this reference manual.

Controlling Access via Mandates

If your company is using the mandates concept, the mandate settings for users and objects in the Alfabet database are also taken into account for responses from the Alfabet RESTful API requesting object data or creating, deleting or updating objects.



For more information about the mandate concept and the required configuration steps see *Implementing the Mandate Capability for a Federated Architecture* in the reference manual *Configuring Alfabet with Alfabet Expand*.

In the payload of the service call, a mandate can be defined for the call with the field `CurrentMandate` as described in the description of service calls in the section [Service Calls and Return Values](#). This setting is optional, but it is recommended to define the mandate in the call. The system checks whether the mandate defined in the call is also specified for the user used to send the service call. The user must either have the mandate assigned or be specified as mandate master. Otherwise the service call will be rejected.

When the mandate concept is implemented, the following rules apply to return of data via calls of the Alfabet RESTful API:

	No CurrentMandate is defined in the service call	A CurrentMandate is defined in the service call
The user is a mandate master and additionally has one or multiple mandates assigned.	The user will see data about all objects regardless of the mandate assignment of the objects in the return calls.	The user will see data about all objects assigned to the mandate defined with <code>CurrentMandate</code> and about all objects with no mandate assigned.
The user has a single mandate assigned and is not the mandate master.	The user will see data about all objects assigned to the mandate assigned to him and about all objects with no mandate assigned.	If the mandate specified with <code>CurrentMandate</code> is assigned to the user, the user will see data about all objects assigned to that mandate and about all objects with no mandate assigned. If the mandate specified with <code>CurrentMandate</code> is not assigned to the user, the service call is rejected with a message that informs about the fact that the <code>CurrentMandate</code> is not assigned to the user
The user has multiple mandates assigned and is not the mandate master.	The user will see data about all objects assigned to the default mandate assigned to him and about all objects with no mandate assigned. The first mandate in alphabetical order of the mandates assigned to the user is the default mandate used.	If the mandate specified with <code>CurrentMandate</code> is assigned to the user, the user will see data about all objects assigned to that mandate and about all objects with no mandate assigned. If the mandate specified with <code>Cur-</code>

	No CurrentMandate is defined in the service call	A CurrentMandate is defined in the service call
		rentMandate is not assigned to the user, the service call is rejected with a message that informs about the fact that the CurrentMandate is not assigned to the user
The user has no mandate assigned and is not a mandate master.	The user will see data about all objects that do not have a mandate assigned.	The service call is rejected with a message that informs about the fact that no mandates are assigned to the user.

If data about an object is requested with a service call to the endpoint `objects` and the mandate settings of the object do not match the mandate settings of the user or the mandate specified in the service call, no data is returned. Instead, two messages about the mandate mismatch are added to the return value:

```
"RejectedObjects": [
  {
    "Id": "",
    "RefStr": "76-3473-0",
    "Message": "Mandate mismatch"
  }
],
"AccessDenied": {
  "76-3473-0": "Mandate mismatch"
}
```

If data of an object shall be updated with a service call to the endpoint `update` or an object shall be deleted with a service call to the endpoint `delete`, the return value also includes the `AccessDenied` field to inform about the mandate mismatch for individual objects.

Controlling Access per Object via Access Permission Concepts

When access permissions per object class have been granted to the user executing the service call to the Alfabet RESTful API as described in the section [Configuring Per Object Class Permissions for Reading From or Writing Into the Alfabet database](#) and the users mandate concepts also permit access as described in the section [Controlling Access Via Mandates](#), write access to objects might nevertheless be rejected because of per object access permissions. The access permission concepts for a user that are implemented in your Alfabet solution are also evaluated for access to objects via the REST API service calls.



For an overview of available access permission concepts in Alfabet and the required configurations to implement them, see the section *Configuring Access Permissions for Alfabet* in the reference manual *Configuring Alfabet with Alfabet Expand*.

The following must be taken into account:

- User profiles can be configured to grant either `ReadWrite` or `ReadOnly` access to Alfabet objects. The configuration of the user profile used for the Alfabet RESTful API must be configured to grant the required access permissions.
- The user must have write access permissions to an object based on for example the authorized user concept or right rules to update or delete object data via a REST API service call.

Usually, a REST API specific user is used to for service calls to the Alfabet RESTful API. This user is not used for creating and updating objects on the Alfabet user interface. Therefore write access permission based for example on the assignment as authorized user or availability of open assignments for the object are not relevant in the context of the Alfabet RESTful API. To grant access to objects for the REST API specific user, you can for example assign the user to relevant authorized user groups or define a right rule to allow write access to objects to the specific user logged in with the specific user profile.

Required Implementation on Client Side

The RESTful service interface on client side must be implemented to send a request for authorization code prior to sending a data request. By default, the authorization code received in the response of the authorization request can be used in data request calls to the Alfabet RESTful API posted within 20 minutes after receiving the authorization code. The time period can be changed for the user in the **User** editor.



For more information about changing the validity period for the authorization code, see [Required Configuration on Server Side](#).

The authorization call must be send to the following URL:

```
URLOfTheAlfabetWebApplication/api/token
```



The specification of the URL is case sensitive.

The method for the call is **POST**.

The `Content-Type` field of the HTTP header must be defined as:

```
Content-Type: application/x-www-form-urlencoded
```

The payload of the request must be defined as:

```
grant_type=password&username=Alfabet&password=AFPUVOIM462LG3EOMY3A6G4T4WNATJI3
```

The answer is a JSON response with the following structure:

```
{
  "token_type": "bearer"
  "access_token": "adf93nfpornpor"
}
```

The calls to the Alfabet RESTful API must be send to the following URL amended with the specification of the service call as described in the following chapters:

`URLOfTheAlfabetWebApplication/api/v2`



The specification of the URL is case sensitive.

The data in the response of the authorization call must be added as authorization key to the header of all requests sent to the Alfabet RESTful API in the following format:

```
Authorization:TypeValue
```

with

Type = the value in the `token_type` field

Value = the value in the `access_token` field



For the answer above, the Authorization would be:

```
Authorization: bearer adf93nfporesopor
```

The authorization key expires after approximately 20 minutes if not otherwise configured in the **User** editor for the user used for key generation. The client interface should be configured to request a new authorization key in regular intervals or prior to each request.

Overview of Access Permissions Required for Each Endpoint

The configuration of access permissions for the Alfabet RESTful services is a complex process to enable establishing of a maximum of security for each use case. To ease configuration for a specific RESTful service targeting the Alfabet RESTful API, the following table lists all available kinds of service calls and the access permissions that must be configured.

Functionality	Endpoint	API Access Options for the named user having the option Has API V2 Access enabled	Attribute to be activated in the relevant class settings of the object class	Additional permission concepts that apply
Exporting Information about the Complete Alfabet Class Model Including Enumerations and Culture Settings	metamodel	Has Meta Model Access		
Exporting Information about	classes	Has Meta Model Access		

Functionality	Endpoint	API Access Options for the named user having the option Has API V2 Access enabled	Attribute to be activated in the relevant class settings of the object class	Additional permission concepts that apply
All or Multiple Classes of the Alfabet Class Model				
Exporting Information about All or Multiple Enumerations in the Alfabet Class Model	enums	Has Meta Model Access		
Exporting Data About Objects with Defined REFSTR Values	objects	Has GetObjectsByRefs Access	Allow Read via Rest API	Mandate settings for user and object
Exporting Information About Objects Found By A Configured Report	objects	Has GetObjectsByReport Access		Mandate settings for user and object Access permission of user and execution permission for REST API for configured report
Exporting Data About Objects Of a Defined Object Class Matching A Filter Definition	objects	Has GetObjectsByFilter Access	Allow Read via Rest API	Mandate settings for user and object
Deleting Objects from the Alfabet database	delete	Has DeleteObjects Access	Allow Write via Rest API	Access permissions of user for object

Functionality	Endpoint	API Access Options for the named user having the option Has API V2 Access enabled	Attribute to be activated in the relevant class settings of the object class	Additional permission concepts that apply
Archiving Objects from the Alfabet database	archive	Has GetObjectsByRefs Access (for all calls) Has DeleteObjects Access (only if objects shall be deleted after archiving)	Allow Read via Rest API (for all calls) Allow Write via Rest API (only if objects shall be deleted after archiving)	Access permissions of user for object
Creating a new Object in the Alfabet database	update	Has CreateObjects Access	Allow Write via Rest API	Access permissions of user for object
Changing the Properties of an existing Object in the Alfabet database	update	Has UpdateObjects Access	Allow Write via Rest API	Access permissions of user for object
Starting an ADIF Import via RESTful Service Call	adifimport	Has ADIFAPIInvocation Access		Execution permission for REST API for ADIF import scheme
Starting an ADIF Export via RESTful Service Call	adifexport	Has ADIFAPIInvocation Access		Execution permission for REST API for ADIF export scheme
Starting a Workflow via REST-	workflow	Has WorkflowAPIInvo-		Execution permission for

Functionality	Endpoint	API Access Options for the named user having the option Has API V2 Access enabled	Attribute to be activated in the relevant class settings of the object class	Additional permission concepts that apply
ful Service Call		ation Access		REST API for workflow template
Regenerating the Password of an Alfabet User	regeneratepassword	Has PasswordAPI Access Has UpdateObjects Access	Allow Read via Rest API for the object class Person Allow Write via Rest API for the object class Person	
Anonymizing User Data For Selected Users	anonymizeuser	Has AnonymizeUser Access		Anonymization must be activated for the object class person.
	idocfilelist	Has IDOC File List Access		
	idocupload	Has IDOC Upload Access		
	idocdownload	Has IDOC Download Access		
Checking Whether the Alfabet components are Running	monitor	Has MonitoringAPI Access		

Chapter 5: Configuring Handling of Date, Time and Number Formats for API Calls

By default, date and time formats have to be defined and are returned in the standard formats defined by your language settings.

The Alfabet Web Application can be configured both to return date and time information in other formats and even to accept other date and time formats in requests. Different sets of date and time format can be configured for concurrent use by different applications that all send requests to the Alfabet RESTful API and use a different date/time format.



Please note however, that for requests to the endpoint `objects` that request results from a configured report with parameters, the parameter settings in the **ReportArgs** field must be defined in the standard formats specified above independent from the date/time format specified for the request.

To define a date and time format for the Alfabet RESTful API, an **API Culture** must be defined for the Alfabet solution with the tool Alfabet Expand. The **API Culture** must then be referenced in the service call to use the defined date, date time and time formats of that API Culture:

- [Configuring a New API Culture](#)
- [Using an API Culture in a Service Call](#)

Configuring a New API Culture

Date and Time definition for the Alfabet RESTful API service calls are defined in the Meta-Model explorer of the tool Alfabet Expand.



For information about how to access and work with Alfabet Expand, see the reference manual *Configuring Alfabet with Alfabet Expand*.



Please consider the following syntax for the specification of date and time formats.

- To specify the 24-hour clock system, use "H" or "HH" for hours.
 - To specify the 12-hour clock system, use "h" or "hh". This setting requires additional specification of AM or PM when a time is written.
 - Use "M" for month and "m" for minute.
- 1) In Alfabet Expand, go to the Meta-Model tab.
 - 2) Right click the **API Cultures** node in the explorer and select **Add New API Culture**. A new API Culture element is added to the explorer and the attributes of the new elements are displayed in the attribute window.
 - 3) In the attribute **API Culture Name**, change the default name of the API Culture element to a unique and meaningful name. The name of the API Culture element is used to reference the API Culture in service calls to the Alfabet RESTful API.

- 4) Select one of the following from the dropdown list in the **Date Format** field to define date and date time formats:
 - **Default:** The default date pattern yyyy-MM-dd and date time pattern yyyy-MM-dd HH:mm:ss.fff is used.
 - **Pattern:** A customer defined pattern is used. If you select this option, two new attributes **Date Pattern** and **Date/Time Pattern** are displayed. Write the required date pattern for date only values into the **Date Pattern** attribute and the required date time pattern into the **Date/Time Pattern** attribute.
 - **Posix:** Dates and date time values are accepted and returned in posix format.
 - **Windows:** The Windows date pattern MM/dd/yyyy and date time pattern MM/dd/yyyy HH:mm is used.
- 5) Select one of the following from the dropdown list in the **Number Format** field to define number formats:
 - **Default:** The default number format is used. The decimal symbol is a dot, the number of decimals is not limited and the group symbol is a comma (e.g. 3,125.987).
 - **Custom:** A customer defined pattern is used. If you select this option, three new attributes **Number Grouping Symbol**, **Number of Decimal Digits** and **Number Decimal Symbol** are displayed. Define the required number format in the new fields. If a field is undefined, the default is used for this setting.
- 6) Select one of the following from the dropdown list in the **Time Format** field to define time formats:
 - **Default:** The default time pattern HH:mm:ss is used.
 - **Pattern:** A customer defined pattern is used. If you select this option, a new attribute **Time Pattern** is displayed. Write the required time pattern into the field.
- 7) In the toolbar, click the **Save**  button to save your changes.

Using an API Culture in a Service Call

API Cultures defined for the Alfabet solution can be used in service calls to the endpoints `objects`, `delete` and `update`. To use an API Culture in a service call, a field `ApiCulture` must be added to the JSON object in the payload of the call. The field value must specify the name of a defined API Culture.



For example the following request for data about objects of the defined REFSTR values returns results with dates defined in Posix format:

```
{
  "ApiCulture": "ApiCulture_Posix",
  "EmptyValues": true,
  "Refs": [ "76-3472-0", "76-3473-0", "76-3474-0" ] }
}
```

This will result in a return value with Posix date formats:

```
{
  "Objects": [
    {
      "ClassName": "Application",
```

```
"RefStr": "76-3473-0",
"Values": {
  "applicationgroups": null,
  "baseapplicationid": null,
  "color": null,
  "creation_date": " 1504696528 ",
  "creation_user": "ALFABET", "description": null,
  ...
}
...
}
]
}
```

Chapter 6: Service Calls and Return Values

Service calls are sending the information about what action to perform in the Alfabet database or what information to send back to the URL of the Alfabet RESTful API of an Alfabet Web Application. The endpoints that are available to access and to change data are predefined and require a specific set of parameters to be sent to the Alfabet RESTful API.

To make a service call, a valid URL consisting of the address of the Alfabet RESTful API and the specification of the request endpoint must be sent to the web server via an HTTP client. The header of the request must contain a valid authorization as described in the chapter above. For some endpoints, the parameters required for execution of the request must be submitted in JSON format in the payload of the request.

A valid URL has the following structure:

```
ServerAddress/api/v2/EndpointName/?Parameters
```

with:

Variable in URL	Required Value
ServerAddress	The web address of the Alfabet Web Application. The specification of the web address is case sensitive.
api/v2	The specification of the Alfabet RESTful API. The current version is <code>api/v2</code> .
EndpointName	The endpoint to be called. Allowed values are: <ul style="list-style-type: none"> <code>metamodel</code> for calls returning information about all classes and cultures in the Alfabet meta-model. For more information see Exporting Information about the Complete Alfabet Class Model Including Enumerations and Culture Settings.

Variable in URL	Required Value
	<ul style="list-style-type: none"> • <code>classes</code> for calls returning information about all or a subset of the classes in the Alfabet meta-model. • <code>enums</code> for calls returning information about all or a subset of enumerations defined in the Alfabet meta-model • <code>objects</code> for calls returning property values of one or multiple defined object in the Alfabet database or the result of a query executed via an Alfabet configured report. • <code>delete</code> for deleting objects and relations from the Alfabet database. • <code>archiveobject</code> for exporting information about selected objects in the Alfabet database in a ZIP file containing the most relevant page views for the objects in HTML format. The endpoint can optionally delete the objects from the Alfabet database after archiving. • <code>update</code> for updating data in the Alfabet database and to create new objects and relations. • <code>regeneratepassword</code> to regenerate the password of selected users in the Alfabet database. • <code>anonymizeuser</code> for anonymizing the data of one or multiple selected users in the Alfabet database. • <code>workflow</code> to start a workflow based on an existing workflow template in the Alfabet database. • <code>adifimport</code> to start an ADIF job based on an ADIF import scheme in the Alfabet database. • <code>adifexport</code> to start an ADIF job based on an ADIF export scheme in the Alfabet database. • <code>monitor</code> to check whether the Alfabet components that might be involved in the execution of a RESTful service request are running and can be accessed.
Parameters	<p>The specification of parameters only applies to the methods <code>metamodel</code>, <code>classes</code> and <code>enums</code>.</p> <p>A parameter is defined as:</p> <p style="text-align: center;"><i>ParameterName=ParameterValue</i></p> <p>If multiple parameters are defined, they are concatenated with an <code>&</code> character:</p> <p style="text-align: center;"><i>ParameterName1=ParameterValue1&ParameterName2=ParameterValue2</i></p>

For the requests of the endpoints `objects` and `update`, the parameters for the request are defined in the payload of the request in JSON format. Details about the calls are given for each request in the following chapters.

Each endpoint provides data in JSON format. When the response includes data of the type `date`, the dates are by default strings of the format `yyyy-mm-dd`. The date format can be changed with a parameter setting in the payload of the request.

The endpoints are described in detail in the following sections, including information about the HTTP method used and the structure of the service call and the return value:

- [Exporting Information about the Complete Alfabet Class Model Including Enumerations and Culture Settings](#)
- [Exporting Information about All or Multiple Classes of the Alfabet Class Model](#)
- [Exporting Information about All or Multiple Enumerations in the Alfabet Class Model](#)
- [Exporting Information About Object Data Stored in the Alfabet database](#)
 - [Exporting Data About Objects with Defined REFSTR Values](#)
 - [Exporting Data About Objects Of a Defined Object Class Matching A Filter Definition](#)
 - [Exporting Information About Objects Found By A Configured Report](#)
 - [Configuring a Report that Can be Called in a RESTful Service Request](#)
 - [Structure of the Service Call](#)
 - [Return Value for the ReportResultDataSet Returning the Data Defined In the Report](#)
 - [Return Value for the ReportResultObjects Returning All Properties of the Objects Found in the Report](#)
 - [Return Value for the ReportResult Returning the REFSTR of the Objects Found in the Report](#)
- [Deleting Objects from the Alfabet database](#)
- [Creating and Updating Object Data in the Alfabet database](#)
 - [Creating a new Object in the Alfabet database](#)
 - [Changing the Properties of an existing Object in the Alfabet database](#)
 - [Creating or Updating a Relation Between Objects in the Alfabet database](#)
- [Archiving Objects from the Alfabet database](#)
- [Regenerating the Password of an Alfabet User](#)

- [Anonymizing User Data For Selected Users](#)
- [Starting a Workflow via RESTful Service Call](#)
- [Starting an ADIF Import via RESTful Service Call](#)
- [Starting an ADIF Export via RESTful Service Call](#)
- [Exporting Information about the Content of the Internal Document Selector](#)
- [Downloading Documents from the Internal Document Selector](#)
- [Uploading Documents to the Internal Document Selector](#)
- [Checking Whether the Alfabet components are Running](#)

Exporting Information about the Complete Alfabet Class Model Including Enumerations and Culture Settings

The endpoint `metamodel` returns information about the definition of all Alfabet object classes, the enumeration and the culture settings in the meta-model.



For detailed information about the structure of the Alfabet meta-model and how data is stored in the Alfabet database tables, see the chapter *The Alfabet Meta-Model in the Alfabet Database* in the reference manual *Alfabet Data Integration Framework* or the reference manual *Alfabet Meta-Model*.

Endpoint name: `metamodel`

HTTP method: `GET`

Service call:

`ServerAddress/api/v2/metamodel/?emptyValues=true`

with the following parameters:

Parameter	Mandatory/Optional	Required Value
-----------	--------------------	----------------

Parameter	Mandatory/Optional	Required Value
<code>emptyValues=true/false</code>	Optional	If <code>emptyValues=true</code> is added to the service call, all relevant attributes of the object classes, enumerations and culture settings are returned even if they are not set. If <code>emptyValues=false</code> is added, only attributes that are set are returned. By default the attribute is set to <code>false</code> .

Header Fields:

`Authorization:TypeValue`



For information about the required Authorization header content, see [Authorization](#).

Return value:

The return value is a JSON object with four fields:

- `alias`: The name of the server alias of the Alfabet Web Application answering the request.
- `cultures`: A JSON list of cultures defined in the Alfabet meta-model.
- `classes`: A JSON list of classes defined in the Alfabet meta-model.
- `enums`: A JSON list of enumerations defined in the Alfabet meta-model.



```
{
  "alias": "Alfabet"
  -"cultures": [...]
  -"classes": [...]
  -"enums": [...]
}
```

The fields `cultures`, `classes` and `enums` all consist of a JSON list of objects. Each object class, enumeration and culture setting in the Alfabet meta-model is an object in the respective JSON list. The attributes of the meta-model object are fields of the JSON object with a name corresponding to the name of the attribute. Attributes specify the technical data about the meta-model object that is used to build the database tables and to process the object data within the Alfabet solution. The list of attributes is limited to the attributes that may be relevant for external applications. For example the date of the last update or tags defined for the object are not exported.

Subordinate objects, like for example enumeration items for enumerations are listed as a field that contains the subordinate objects as a JSON list of objects.

The following information is provided:

Object Type in the Alfabet meta-model	Information provided in the response	Example
Culture Settings	<p>Each culture setting in the Alfabet meta-model is a JSON object in the JSON list of the field <code>cultures</code>.</p> <p>Each relevant attribute defined for the culture setting in the Alfabet meta-model is defined as a field in the JSON object. The field names are similar to the names of the attributes.</p>	<pre>"cultures": [7] 0: { "BaseCulture": 1031 "DateFormatCulture": 1031 "NumberFormatCulture": 1031 "HelpCulture": 1031 "IsDefault": false "IsInstanceTranslation": true "Picture": "Flag_German" "TimePattern": "HH:mm:ss" "MeasurementUnit": "Metric" } ...</pre>
Object Classes	<p>Each object class in the Alfabet meta-model that is visible in the Classes explorer of the Metamodel tab of the configuration tool Alfabet Expand is a</p>	<pre>"classes": [266]</pre>

Object Type in the Alfabet meta-model	Information provided in the response	Example
	<p>JSON object in the JSON list of the field <code>classes</code>.</p> <p>Information about object classes includes the following:</p> <ul style="list-style-type: none"> Attributes of the object class <p>The relevant attributes defined for the object class in the Alfabet meta-model are each defined as a field in the JSON object. The field names are similar to the names of the attributes.</p> <p>Please note that object classes having the attribute Automatically Managed set to <code>true</code> are object classes that shall only be changed by mechanisms triggered by the Alfabet software components and must not be changed by any third party component.</p> Object class properties <p>For each Alfabet object class a database table exists in the Alfabet database. Each object class property corresponds to a column in the database table, that means that the object class properties store the information about individual objects of the class. All object class properties that are visible in the Classes explorer of the Metamodel tab of the configuration tool Alfabet Expand are included into the field <code>Properties</code> as a JSON list of object, each JSON object representing a property and the fields of the JSON object representing a subset of the attributes of the property that might be relevant for external processes.</p> <p>Please note that object class properties having the attribute Automatically Managed set to <code>true</code> are properties that shall only be changed by the Alfabet software components and must not be changed by any third party component.</p> Object class stereotypes 	<pre> 0: { "Name": "Domain" "Id": 359 "Comment": "" "Caption": "Domain" "AutomaticallyManaged": false "TechName": "DOMAIN" "Audit": true "HasMandates": true "Hint": "" "IDPrefix": "DOM" "Stereotypes": [5] 0: { "Name": "Area" "Caption": "Area" "CaptionPlural": "Areas" "comment": "" "HasMandates": false } ... "Properties": [31] 0: { "Name": "ID" "Guid": </pre>

Object Type in the Alfabet meta-model	Information provided in the response	Example
	<p>If stereotypes are configured for the object class, the stereotypes are included as a JSON list of objects in the field <code>Stereotypes</code> of the class object. Each attribute of the stereotype is exported as a field in the JSON object.</p> <ul style="list-style-type: none"> Class Keys <p>A class key with an attribute Unique set to <code>True</code> specifies one or a combination of object class properties that must be unique for that class. If the data does not fulfill the requirements specified in the class key definition, the object cannot be created in the database. The Alfabet meta-model also allows class keys to be specified that do not require uniqueness. In this case, the class key attribute Unique will be set to <code>False</code>. The purpose of such a class key is to speed up the search functionality by creating an index for each class key. In the return value, each class key is an object and listed in an array of objects in the field <code>"unique keys"</code> in the object class object.</p> 	<pre> "6736AEF52C416C94D161FAABAF3D7F" "TechName": "ID" "Caption": "ID" "Comment": "" "Hint": "" "Alias": "" "Type": "String" "AutomaticallyManaged": false "DefaultValue": null "Validator": "" "EnumInfo": "" } ... "Keys": [3] 0: { "Name": "Domain_Key1" "techName": "C359_K1" "Content": "BelongsTo" "Unique": false "IsCaseIns": false "Descending": false "IsActivated": true } ... </pre>

Object Type in the Alfabet meta-model	Information provided in the response	Example
		}
Enumerations	<p>Each enumeration in the Alfabet meta-model is a JSON object in the JSON list of the field <code>enums</code>.</p> <p>Information about enumeration includes the following:</p> <ul style="list-style-type: none"> Attributes of the enumeration <p>Each relevant attribute defined for the enumeration in the Alfabet meta-model is defined as a field in the JSON object. The field names are similar to the names of the attributes.</p> Enumeration items and their attributes <p>The enumeration items defined for the enumeration are listed in the field <code>Items</code>. The field <code>Items</code> provides a JSON list of objects. Each enumeration item is an object in that list and the fields of the object return the relevant attributes of the enumeration item. The field names are similar to the names of the attributes.</p> 	<pre> "enums": [90] 0: { "Name": "AlfaDocCategory" "Guid": "6E7E549625034B788C5725F2541BD412" "Comment": "" "Hint": "" "HelpFile": "Enum_AlfaDocCategory.html" "Items": [7] 0: { "Value": "" "Comment": "" "Hint": "" } 1: { "Value": "Manual" "Comment": "" "Hint": "" } ... } </pre>

Exporting Information about All or Multiple Classes of the Alfabet Class Model

The endpoint `classes` returns information about the definition of all or a subset of the Alfabet object classes in the Alfabet meta-model.



For detailed information about the structure of the Alfabet meta-model and how data is stored in the Alfabet database tables, see the chapter *The Alfabet Meta-Model in the Alfabet Database* in the reference manual *Alfabet Data Integration Framework* or the reference manual *Alfabet Meta-Model*.

Endpoint name: `classes`

HTTP method: `GET`

Service call:

```
ServerAddress/api/v2/classes/?names=Domain,Application&emptyValues=true
```

with the following parameters:

Parameter	Mandatory/Optional	Required Value
<code>names= ClassName, ClassName</code>	Optional	A comma separated list of object classes for that information shall be returned. The object class must be defined by the value of the Name attribute of the object class. If the parameter is not added to the call, information about all object classes in the Alfabet meta-model is returned.
<code>emptyValues=true/false</code>	Optional	If <code>emptyValues=true</code> is added to the service call, all relevant attributes of the object classes are returned even if they are not set. If <code>emptyValues=false</code> is added, only attributes that are set are returned. By default the attribute is set to <code>false</code> .

Header Fields:

```
Authorization:TypeValue
```



For information about the required Authorization header content, see [Authorization](#).

Return value:

The return value is a JSON object with two fields:

- `alias`: The name of the server alias of the Alfabet Web Application answering the request.
- `classes`: A JSON list of classes defined in the Alfabet meta-model.



```
{
  "alias": "Alfabet"
  "classes": [...]
}
```

The field `classes` consists of a JSON list of objects. Each object class in the Alfabet meta-model is an object in the JSON list. The attributes of the meta-model object class are fields of the respective JSON object. The name of the field corresponds to the name of the attribute. Attributes specify the technical data about the meta-model object that is used to build the database tables and to process the object data within the Alfabet solution. The list of attributes is limited to the attributes that may be relevant for external applications. For example the date of the last update or tags defined for the object are not exported.

Subordinate objects, like for example stereotypes defined for the class are listed as a field that contains the subordinate objects as a JSON list of objects.

The following information is provided:

If the parameter `names` is not specified in the request, each object class in the Alfabet meta-model that is visible in the **Classes** explorer of the **Metamodel** tab of the configuration tool Alfabet Expand is a JSON object in the JSON list of the field `classes`. The classes are listed in alphabetical order.

If the parameter `names` is specified in the request, the list of classes contains only the classes defined in the `names` parameter in the order specified in the `names` parameter.

Information about object classes includes the following:

```
"classes": [266]
  0: {
    "Name": "Domain"
    "Id": 359
    "Comment": ""
    "Caption": "Domain"
    "AutomaticallyManaged": false
```

- **Attributes of the object class**

The relevant attributes defined for the object class in the Alfabet meta-model are each defined as a field in the JSON object. The field names are similar to the names of the attributes.

Please note that object classes having the attribute **Automatically Managed** set to `true` are object classes that shall only be changed by mechanisms triggered by the Alfabet software components and must not be changed by any third party component.

- **Object class properties**

For each Alfabet object class a database table exists in the Alfabet database. Each object class property corresponds to a column in the database table, that means that the object class properties store the information about individual objects of the class. All object class properties that are visible in the **Classes** explorer of the **Metamodel** tab of the configuration tool Alfabet Expand are included into the field `Properties` as a JSON list of object, each JSON object representing a property and the fields of the JSON object representing a subset of the attributes of the property that might be relevant for external processes.

Please note that object class properties having the attribute **Automatically Managed** set to `true` are properties that shall only be changed by the Alfabet software components and must not be changed by any third party component.

- **Object class stereotypes**

If stereotypes are configured for the object class, the stereotypes are included as a JSON list of objects in the field `Stereotypes` of the class object. Each attribute of the stereotype is exported as a field in the JSON object.

- **Class Keys**

A class key with an attribute **Unique** set to `True` specifies one or a combination of object class properties that must be unique for that class. If the data does not fulfill the requirements specified in the class key definition, the object cannot be created in the database. The Alfabet meta-model also allows class keys to be specified that do not require uniqueness. In this case, the class key attribute **Unique** will be set to `False`. The purpose of such

```
"TechName": "DOMAIN"
"Audit": true
"HasMandates": true
"Hint": ""
"IDPrefix": "DOM"
"Stereotypes": [5]
  0: {
    "Name": "Area"
    "Caption": "Area"
    "CaptionPlural": "Areas"
    "comment": ""
    "HasMandates": false
  }
  ...
"Properties": [31]
  0: {
    "Name": "ID"
    "Guid":
    "6736AEFBF52C416C94D161FAABAF3D7F"
    "TechName": "ID"
    "Caption": "ID"
    "Comment": ""
    "Hint": ""
    "Alias": ""
    "Type": "String"
    "AutomaticallyManaged": false
    "DefaultValue": null
    "Validator": ""
```

a class key is to speed up the search functionality by creating an index for each class key. In the return value, each class key is an object and listed in an array of objects in the field "unique keys" in the object class object.

```

    "EnumInfo": ""
  }
  ...
  "Keys": [3]
  0: {
    "Name": "Domain_Key1"
    "techName": "C359_K1"
    "Content": "BelongsTo"
    "Unique": false
    "IsCaseIns": false
    "Descending": false
    "IsActivated": true }
  ...
}

```

Exporting Information about All or Multiple Enumerations in the Alfabet Class Model

The endpoint `enums` returns information about the definition of all or a subset of the enumerations in the Alfabet meta-model. Enumerations define a predefined set of values that are allowed to be set for an object class property.



For detailed information about the structure of the Alfabet meta-model and how data is stored in the Alfabet database tables, see the chapter *The Alfabet Meta-Model in the Alfabet Database* in the reference manual *Alfabet Data Integration Framework* or the reference manual *Alfabet Meta-Model*.

Endpoint name: `enums`

HTTP method: `GET`

Service call:

```
ServerAdress/api/v2/enums/?names=Enum1,Enum2&emptyValues=true
```

with the following parameters:

Parameter	Mandatory/Optional	Required Value
names= EnumName, EnumName	Optional	A comma separated list of enumeration for that information shall be returned. The enumeration must be defined by the value of the Name attribute of the enumeration. If the parameter is not added to the call, information about all enumerations in the Alfabet meta-model is returned.
emptyValues=true/false	Optional	If <code>emptyValues=true</code> is added to the service call, all relevant attributes of the enumeration and enumeration items are returned even if they are not set. If <code>emptyValues=false</code> is added, only attributes that are set are returned. By default the attribute is set to <code>false</code> .

Header Fields:

```
Authorization:TypeValue
```



For information about the required Authorization header content, see [Authorization](#).

Return value:

The return value is a JSON object with two fields:

- `alias`: The name of the server alias of the Alfabet Web Application answering the request.
- `classes`: A JSON list of classes defined in the Alfabet meta-model.



```
{
  "alias": "Alfabet"
  -"enum": [...]
}
```

```
}

```

The field `classes` consists of a JSON list of objects. Each enumeration in the Alfabet meta-model is an object in the JSON list. The attributes of the enumeration are fields of the respective JSON object. The name of the field corresponds to the name of the attribute. Attributes specify the technical data about the meta-model object that is used to build the database tables and to process the object data within the Alfabet solution. The list of attributes is limited to the attributes that may be relevant for external applications. For example the date of the last update or tags defined for the object are not exported.

Subordinate objects, like for example enumeration items defined for the enumeration are listed as a field that contains the subordinate objects as a JSON list of objects.

The following information is provided:

If the parameter `names` is not specified in the request, each enumeration in the Alfabet meta-model that is visible in the **Classes** explorer of the **Metamodel** tab of the configuration tool Alfabet Expand is a JSON object in the JSON list of the field `enums`. The enumerations are listed in alphabetical order.

If the parameter `names` is specified in the request, the list of enumerations contains only the enumerations defined in the `names` parameter in the order specified in the `names` parameter.

Information about object classes includes the following:

- **Attributes of the enumeration**

Each relevant attribute defined for the enumeration in the Alfabet meta-model is defined as a field in the JSON object. The field names are similar to the names of the attributes.

- **Enumeration items and their attributes**

The enumeration items defined for the enumeration are listed in the field `Items`. The field `Items` provides a JSON list of objects. Each enumeration item is an object in that list and the fields of the object return the relevant attributes of the enumeration item. The field names are similar to the names of the attributes.

```
"enums": [90]
  0: {
    "Name": "AlfaDocCategory"
    "Guid":
      "6E7E549625034B788C5725F2541BD412"
    "Comment": ""
    "Hint": ""
    "HelpFile":
      "Enum_AlfaDocCategory.html"
    "Items": [7]
      0: {
        "Value": ""
        "Comment": ""
        "Hint": ""
      }
      1: {
        "Value": "Manual"
        "Comment": ""
        "Hint": ""
      }
    
```

```
}  
...  
}
```

Exporting Information About Object Data Stored in the Alfabet Database

The endpoint `objects` returns the information stored about one or multiple objects in a database table of the Alfabet database. There are three methods to select the objects for that information is returned by this endpoint:

- **Objects By References**

This method requires that the value of the `REFSTR` property of the object is known. The `REFSTR` is a unique identifier for objects in the Alfabet database. It has to be defined in the payload of the request to export all property values set for the object.

- **Objects By Filter**

The method returns all property values set for all objects of a specified object class matching search conditions that are defined in the service call. The search filter in the service call can only search in object class properties returning a text, that means object class properties of the type `Text` or `String`.

- **Objects By Report**

The payload of the request can point to a configured report of the type `NativeSQL` or `Query` in the Alfabet database. The return value includes all information included in the dataset of the report or a subset thereof.

The following information is available:

- [Exporting Data About Objects with Defined REFSTR Values](#)
- [Exporting Data About Objects Of a Defined Object Class Matching A Filter Definition](#)
- [Exporting Information About Objects Found By A Configured Report](#)
 - [Configuring a Report that Can be Called in a RESTful Service Request](#)
 - [Structure of the Service Call](#)
 - [Return Value for the ReportResultDataSet Returning the Data Defined In the Report](#)

- [Return Value for the ReportResultObjects Returning All Properties of the Objects Found in the Report](#)
- [Return Value for the ReportResult Returning the REFSTR of the Objects Found in the Report](#)

Exporting Data about Objects with Defined REFSTR Values

Endpoint name: `objects`

HTTP method: `POST`

Service call:

```
ServerAddress/api/v2/objects
```

Header Fields:

```
Authorization:TypeValue
```



For information about the required Authorization header content, see [Authorization](#).

```
Content-Type: application/json; charset=utf-8
```

Payload

The payload is a JSON object with the following structure:

```
{  
  "CurrentProfile": "UserProfile1",  
  "CurrentMandate": "Mandate1",  
  "ApiCulture": "ApiCulture_Posix",  
  "Language": 1033,  
  "DataCulture": "de-DE",  
  "EmptyValues": true,  
}
```

```
"Refs": [ "76-2518-0", "76-3246-0", "405-3-0" ]
}
```

It may have the following fields:

Field	Mandatory/Optional	Required Value
"Refs":[" RefstrOfObject "," RefstrOfObject "]	Mandatory	An array containing one or multiple values of the <code>REFSTR</code> property of objects in the Alfabet database. The response will include information about the objects with the objects specified by their <code>REFSTR</code> .
"EmptyValues":true/false	Optional	If <code>EmptyValues</code> is set to true, all object class properties of the object are returned even if they are not set. If <code>EmptyValues</code> is set to false, only object class properties of the object that are set are returned. By default the attribute is set to false.
"CurrentProfile": " UserProfileName "	Mandatory	The name of a user profile assigned to the Alfabet user sending the call that shall be used to access Alfabet. The user profile is required to evaluate whether the user that is sending the service call is allowed to read data about an object. For details about the access permissions that depend on the user profile, see the section Authorization .
"CurrentMandate": " MandateName "	Optional	This field is only relevant if access to objects is controlled by the mandate concept for federated architectures implemented in Alfabet. The field must specify the name of a mandate assigned to the Alfabet user sending the call. For details about the consideration of mandate settings in REST API service calls, see the section Controlling Access Via Mandates .
"Language": " LanguageLCID "	Optional	This field is only relevant if instance translation is used in your Alfabet application. The field has to specify the language code (LCID) decimal of the culture for that translations are available for the name and description of the

Field	Mandatory/Optional	Required Value
		<p>object in the Alfabet database. By default the standard language of the database is returned.</p> <p>Please note that no values are returned when an instance translation is not available for an object class property, even if a value is provided in the standard language.</p> <p>To return all values in the original values and add translations in a second language, the parameter <code>DataCulture</code> shall be used instead of the parameter <code>Language</code>. <code>Language</code> supersedes <code>DataCulture</code>. If both fields are available, only <code>Language</code> is used.</p>
"DataCulture": " LanguageCode "	Optional	<p>This field is only relevant if instance translation is used in your Alfabet application. The field has to specify the language code of the culture for that translations are available for the name and description of the object in the Alfabet database (for example <code>de-DE</code> for German), or <code>ALL</code> to return translations for all languages that provide translation. Translations are added to the return value in addition to the values in the original language in a separate field <code>Translations</code>.</p> <p>The field <code>Language</code> supersedes <code>DataCulture</code>. If both fields are available, only <code>Language</code> is used and <code>DataCulture</code> is ignored.</p>
"ApiCulture": "APICultureName"	Optional	<p>This field specifies an API Culture that was defined in Alfabet Expand as part of the configuration of the Alfabet meta-model. The API Culture allows to define data, date/time and time patterns as well as number formats that deviate from the Alfabet standard. If this field is added to the request, dates, times and numbers are written into the return value in the formats specified in the API Culture definition. For more information, see the section Configuring Handling of Date, Time and Number Formats For API Calls.</p>

Return value:

The return value is a JSON object with two fields:

- **Count:** The number of objects found in the database for that data is returned. The count may differ from the number of defined `REFSTR` values in the request: if an object is not found because the `REFSTR` defined in the request does not exist in the Alfabet database, the object is not included in the count and in the returned object data.
- **Objects:** A JSON list of objects, each object representing a database object for that data is returned.



```
{
  "Objects": [...],
  "Count": 3
}
```

The field `Objects` contains one JSON object per database object found for the call. The object contains the following fields:

- **ClassName:** The name of the Alfabet object class the object belongs to.
- **RefStr:** The value of the `REFSTR` of the object.
- **Values:** A JSON object that includes all information defined about the object in the database via object class properties. Each property is a field with a name corresponding to the property name. The value of the field informs about the value of the property. The return value for a property depends on the value type of the property:

Property Type	Return value
String, Text	The string, text is returned as string (in inverted commas): <pre>"description": "This application manages customer relations."</pre>
StringArray	All selected options of the string array are returned in one string: <pre>"subcategories": "APP_SecurityAssessment APP_CloseSubworkflow"</pre>
Boolean	Boolean values are 1 for true, 0 for false or <code>null</code> if the property is not set:

Property Type	Return value
	<code>"variant": 1</code>
Real, Integer	Real and integer values are returned as number: <code>"xpos": 485.4</code>
Date, DateTime	Date and date time information is both returned as date strings of the format yyyy-mm-dd: <code>"last_update": "2012-02-21"</code>
URL	URLs are returned as string (in inverted commas): <code>"url": "http://company.com/ReportServer"</code>
Reference	The REFSTR of the object the reference is targeting is returned: <code>"ictobject": "26-608-0"</code>
ReferenceArray	<p>For reference arrays, the REFSTRs of all objects that the current object references with this property are listed in an array:</p> <pre>"applicationgroups": [3] 0: "95-38-0" 1: "95-8-0" 2: "95-9-0"</pre> <p>The return value is independent from the setting of the attribute RefSupport of the property. If the attribute RefSupport of the property is set to <code>false</code>, the REFSTR value of the reference targets are directly stored in the database table of the object class in a column for the property. If the attribute RefSupport is set to <code>true</code>, the references are stored in the RELATIONS table. This difference is not visible in the return value for the <code>objects</code> endpoint.</p> <p>If information about the objects referenced by the current object shall be included into the return value, you can use the method to find</p>

Property Type	Return value
	objects via a configured report that is also available for the endpoint <code>objects</code> .

- **Translations:** This field is only available if the field `DataCulture` is defined in the payload of the service call. The field contains a JSON list of object with one object per data culture. Each object has a field `DataCulture` that returns the language for that translations are returned and one field `Values` that is a JSON object with a field for each translated property. The value of the field informs about the translated value of the property.
- **GenericAttributes:** If attributes are defined for the object via the object class `GenericAttribute`, all generic attributes with the object class property `Owner` set to the current object are listed in this field as a JSON list of objects. Each object in the list returns one object of the class `GenericAttribute`, with the fields `Refstr`, `Values` and `Translations` that return data about the generic attribute in the same format as returned for the main object.

Exporting Data about Objects of a Defined Object Class Matching a Filter Definition

Endpoint name: `objects`

HTTP method: `POST`

Service call:

`ServerAdress/api/v2/objects`

Header Fields:

`Authorization:TypeValue`



For information about the required Authorization header content, see [Authorization](#).

`Content-Type: application/json; charset=utf-8`

Payload

The payload is a JSON object with the following structure:

```

{
  "CurrentProfile": "UserProfile1",
  "CurrentMandate": "Mandate1",
  "ApiCulture": "ApiCulture_Posix",
  "Language": 1033,
  "DataCulture": "de-DE",
  "EmptyValues": true,
  "Class": "Application",
  "Limit": "80"
  "FilterTextProperties": [
    { "Name": "Content*", "ShortName": "CMS*" },
    { "Name": "Document*" }
  ]
}

```

It may have the following fields:

Field	Mandatory/Optional	Required Value
"Class": "ObjectClassName"	Mandatory	The value of the Name attribute of the Alfabet object class for that object data shall be returned. Only one object class can be defined for a service call.
"FilterTextProperties": [{ "PropertyName": "FilterValue", "PropertyValue": "FilterValue" }, { "PropertyName": "FilterValue", "PropertyValue": "FilterValue" }]	Mandatory	<p>An array containing the filter definition. The array must contain at least one JSON object with one field. Each field defines a filter condition with the syntax:</p> <pre>"ObjectClassPropertyName": "SearchCondition"</pre> <p>Please note the following about the filter definitions:</p>

Field	Mandatory/Optional	Required Value
		<ul style="list-style-type: none"> The search filter in the service call may only include object class properties returning a text, that means object class properties of the type <code>Text</code> or <code>String</code>. Asterisks can be used as wildcard in the search strings. If the search string is empty, the search returns only objects for that this object class property is not set. Each JSON object in the filter array can have multiple fields. The fields are evaluated with an AND condition. For example the filter <pre>{ "Name": "Content*", "ShortName": "CMS*" }</pre> finds objects with a name starting with <code>Content</code> AND a short name starting with <code>CMS</code>. The filter array can include multiple JSON objects. The filter definitions in different JSON objects are evaluated with an OR condition. For example the filter <pre>{ "Name": "Content*" }, { "ShortName": "CMS*" }</pre> finds objects with a name starting with <code>Content</code> OR a short name starting with <code>CMS</code>.
<code>"Limit":Number</code>	Optional	<p>This field defines the maximum number of objects for that the response of the call shall return data. This value may be set to limit data return for filter definitions returning a high number of objects. In combination with the field <code>Offset</code> this field can be used to fetch the result of a big search in multiple steps.</p> <p>If <code>Limit</code> is not set, a maximum number of 1000 result data sets of the configured report are returned.</p>

Field	Mandatory/Optional	Required Value
"Offset":Number	Optional	This field defines the start position for returning values in the result data set of the configured report. For example if the <code>Limit</code> field is set to 20 and the <code>Offset</code> is set to 10, the results listed in row 11 to 30 in the tabular output of the configured report are returned in the service call. By default, the start position is the first row in the data set with the number 0.
"EmptyValues":true/false	Optional	If <code>EmptyValues</code> is set to true, all object class properties of the object are returned even if they are not set. If <code>EmptyValues</code> is set to false, only object class properties of the object that are set are returned. By default the attribute is set to false.
"CurrentProfile": " UserProfileName "	Mandatory	The name of a user profile assigned to the Alfabet user sending the call that shall be used to access Alfabet. The user profile is required to evaluate whether the user that is sending the service call is allowed to read data about an object. For details about the access permissions that depend on the user profile, see the section Authorization .
"CurrentMandate": " MandateName "	Optional	This field is only relevant if access to objects is controlled by the mandate concept for federated architectures implemented in Alfabet. The field must specify the name of a mandate assigned to the Alfabet user sending the call. For details about the consideration of mandate settings in REST API service calls, see the section Controlling Access Via Mandates .

Field	Mandatory/Optional	Required Value
"Language": "LanguageLCID "	Optional	<p>This field is only relevant if instance translation is used in your Alfabet application. The field has to specify the language code (LCID) decimal of the culture for that translations are available for the name and description of the object in the Alfabet database. By default the standard language of the database is returned.</p> <p>Please note that no values are returned when an instance translation is not available for an object class property, even if a value is provided in the standard language.</p> <p>To return all values in the original values and add translations in a second language, the parameter <code>DataCulture</code> shall be used instead of the parameter <code>Language</code>. <code>Language</code> supersedes <code>DataCulture</code>. If both fields are available, only <code>Language</code> is used.</p>
"DataCulture": " LanguageCode "	Optional	<p>This field is only relevant if instance translation is used in your Alfabet application. The field has to specify the language code of the culture for that translations are available for the name and description of the object in the Alfabet database (for example <code>de-DE</code> for German), or <code>ALL</code> to return translations for all languages that provide translation. Translations are added to the return value in addition to the values in the original language in a separate field <code>Translations</code>.</p> <p>The field <code>Language</code> supersedes <code>DataCulture</code>. If both fields are available, only <code>Language</code> is used and <code>DataCulture</code> is ignored.</p>

Field	Mandatory/Optional	Required Value
"ApiCulture": "APICultureName"	Optional	This field specifies an API Culture that was defined in Alfabet Expand as part of the configuration of the Alfabet meta-model. The API Culture allows to define data, date/time and time patterns as well as number formats that deviate from the Alfabet standard. If this field is added to the request, dates, times and numbers are written into the return value in the formats specified in the API Culture definition. For more information, see the section Configuring Handling of Date, Time and Number Formats For API Calls .

Return value:

The return value is a JSON object with three fields:

- **Count:** The number of objects found in the database for that data is returned.
- **RejectedObjects:** The objects that were found but for that no data was returned because of missing access permissions. The field contains a JSON list of objects with one JSON object for each rejected Alfabet object. The object has a field `RefStr` that returns the `REFSTR` of the Alfabet object and a field `Message` that gives information about the rejection.
- **Objects:** A JSON list of objects, each object representing a database object for that data is returned.



```
{
  "Objects": [...],
  "Count": 3,
  "RejectedObjects": [Array[0]]
}
```

The field `Objects` contains one JSON object per database object found for the call. The object contains the following fields:

- **ClassName:** The name of the Alfabet object class the object belongs to.

- **RefStr**: The value of the `REFSTR` of the object.
- **Values**: A JSON object that includes all information defined about the object in the database via object class properties. Each property is a field with a name corresponding to the property name. The value of the field informs about the value of the property. The return value for a property depends on the value type of the property:

Property Type	Return value
String, Text	The string, text is returned as string (in inverted commas): <code>"description": "This application manages customer relations."</code>
StringArray	All selected options of the string array are returned in one string: <code>"subcategories": "APP_SecurityAssessment APP_CloseSubworkflow"</code>
Boolean	Boolean values are 1 for true, 0 for false or <code>null</code> if the property is not set: <code>"variant": 1</code>
Real, Integer	Real and integer values are returned as number: <code>"xpos": 485.4</code>
Date, DateTime	Date and date time information is both returned as date strings of the format <code>yyyy-mm-dd</code> : <code>"last_update": "2012-02-21"</code>
URL	URLs are returned as string (in inverted commas): <code>"url": "http://company.com/ReportServer"</code>

Property Type	Return value
Reference	<p>The REFSTR of the object the reference is targeting is returned:</p> <pre>"ictobject": "26-608-0"</pre>
ReferenceArray	<p>For reference arrays, the REFSTRs of all objects that the current object references with this property are listed in an array:</p> <pre>"applicationgroups": [3] 0: "95-38-0" 1: "95-8-0" 2: "95-9-0"</pre> <p>The return value is independent from the setting of the attribute RefSupport of the property. If the attribute RefSupport of the property is set to <code>false</code>, the REFSTR value of the reference targets are directly stored in the database table of the object class in a column for the property. If the attribute RefSupport is set to <code>true</code>, the references are stored in the RELATIONS table. This difference is not visible in the return value for the <code>objects</code> endpoint.</p> <p>If information about the objects referenced by the current object shall be included into the return value, you can use the method to find objects via a configured report that is also available for the endpoint <code>objects</code>.</p>

- **Translations:** This field is only available if the field `DataCulture` is defined in the payload of the service call. The field contains a JSON list of object with one object per data culture. Each object has a field `DataCulture` that returns the language for that translations are returned and one field `Values` that is a JSON object with a field for each translated property. The value of the field informs about the translated value of the property.
- **GenericAttributes:** If attributes are defined for the object via the object class `GenericAttribute`, all generic attributes with the object class property `Owner` set to the current object are listed in this field as a JSON list of objects. Each object in the list returns one object of the class `GenericAttribute`, with the fields `Refstr`, `Values` and `Translations` that return data about the generic attribute in the same format as returned for the main object.

Exporting Information about Objects Found by a Configured Report

A configured report returning a tabular dataset that is configured via the configuration tool Alfabet Expand and stored in the Alfabet database can be used to export all or a subset of the data available about objects in the Alfabet database. This method returns data about all objects found as base objects of the report.



For Alfabet queries, the base objects in the reports are the objects of the object class defined in the `FIND` clause. In native SQL queries, the first column of the result set is not displayed in the query results. It must specify the `REFSTR` of the object class selected as base class.

Please note that the base class can be changed to any other class for that data is added to the result data set via the instruction `SetRowReference`.

The report can be used by the objects endpoint of the Alfabet RESTful API to deliver the following data about the objects found by the query of the report:

- The response returns the data that is included in the report. The advantages of this method are the following:
 - The subset of objects returned can depend on a specific parameter, like for example all applications in an application group or all components that are having a defined indicator set to a specific value.
 - The information returned about the objects is configurable. That means that data can be provided in the way and with the field names defined via the report. For example a reference to another object is stored in the database table of the object in a column with the technical name of the property and the value set to the `REFSTR` of the referenced object. The dataset in the query can return this information in a column of a customer defined name with the name of the referenced object instead of the `REFSTR`.
- The response returns information about all property values for all base objects found in the report independent from the data defined in the dataset of the report. This method delivers the same result per object as the method finding objects by `REFSTR`. The advantages of this method are the following:
 - The `REFSTR` of the objects must not be known to the client application when sending the request.
 - The subset of objects returned can depend on a specific parameter, like for example all applications in an application group or all components that are having a defined indicator set to a specific value.
- The response returns all `REFSTR` values of all base objects found in the report. This kind of response is useful if the `REFSTR` values of a defined group of objects are required as input for example for external applications that shall provide links that open the Alfabet user interface. For more information see [Accessing the Alfabet User Interface From the External Application](#).

For the execution of this endpoint methods, a configured report must be available before executing the RESTful service call. The required configuration and the call are described in the following:

- [Configuring a Report that Can be Called in a RESTful Service Request](#)
- [Structure of the Service Call](#)

- [Return Value for the ReportResultDataSet Returning the Data Defined In the Report](#)
- [Return Value for the ReportResultObjects Returning All Properties of the Objects Found in the Report](#)
- [Return Value for the ReportResult Returning the REFSTR of the Objects Found in the Report](#)

Configuring a Report That Can Be Called in a RESTful Service Request

The configuration of configured report and the underlying queries in Alfabet Expand is described in detail in the reference manual *Configuring Alfabet with Alfabet Expand* in the chapters *Configuring Reports* and *Defining Queries*. This information is not repeated here. The following description is limited to additional settings required for the report.

The following special requirements apply to the settings of the attributes of the report:

- The attribute **Applicable for REST API** of the configured report must be set to `True`.
- The report must be of the **Type** `Query` or `NativeSQL`.

The following special requirements apply to the query definition:

- SQL queries must return a `REFSTR` value as first argument. It is possible to create a report based on native SQL returning data without coupling the data to a base object, for example by setting the first argument in the `SELECT` statement to `NULL`. The return value of the Alfabet RESTful service call is structuring the returned data by base object found by in the report. Therefore this kind of report would result in no objects found. As a result, no data is displayed unless the instruction `SetRowReference` is used to define the base object using another row in the dataset to identify the base objects.
- Alfabet query language parameters can be used in Alfabet queries and native SQL queries. If the configured report contains parameter definitions, it is not required to define a **Report View** with filter definitions. The parameters are directly set in the service call.
- Alfabet query language instructions can be used in Alfabet queries and native SQL queries. The dataset after execution of the instructions is used for the generation of the return values. For example if the `JoinColumns` instruction is used in the configured report, the data joined in one column will also be displayed in one field of the result JSON object. The instruction `SetRowReference` alters the base object of the report data set and also for the return value of the service call.
- The configured report must return a simple table. Grouped datasets are not processed correctly. Data for subordinate levels is ignored.

Please note that the user that is used in the authorization of the RESTful service request must have a user profile assigned that has access permissions to the configured report. If none of the user profiles assigned to the user allows access to the configured report, no data is returned by the service request.



For information about configuring access permissions for configured reports, see the chapter *Defining and Managing User Access to Configured Reports* in the reference manual *User and Solution Administration*. While the access permissions for a configured report on the Alfabet user interface depend on multiple different factors, the access permissions for access to the configured report via the RESTful service API only depend on the user profile related access permissions defined for the configured report.

Structure of the Service Call

Endpoint name: `objects`

HTTP method: `POST`

Service call:

```
ServerAddress/api/v2/objects
```

Header Fields:

```
Authorization:TypeValue
```



For information about the required Authorization header content, see [Authorization](#).

```
Content-Type: application/json; charset=utf-8
```

Payload

The payload is a JSON object with the following structure:

```
{
  "CurrentProfile": "UserProfile1",
  "CurrentMandate": "Headquarters",
  "Language": 1033,
  "DataCulture": "fr-FR",
```

```

"ApiCulture": "APICultureReporting",
"EmptyValues": true,
"Report": "Report_1",
"ReportResult": "DataSet",
"Limit": 500,
"Offset": 250,
"ReportArgs":
{
  "arg1": "val1",
  "arg2": "val2",
}
}

```

It may have following fields:

Field	Mandatory/Optional	Required Value
"Report": " ReportName "	Mandatory	The name of the configured report that the call is reading data from. The name of the configured report is the value of the property Name of the configured report.
ReportResult: "DataSet/References/Objects"	Mandatory	ReportResult must be one of the following: <ul style="list-style-type: none"> • DataSet to return all information included in the report. • References to return a list of REFSTR of the objects found in the report • Objects to return all property values set for all objects found in the report
"Limit": Number	Optional	This field defines the maximum number of objects for that the response of the call shall return data. This value may be set to limit data return for reports including a

Field	Mandatory/Optional	Required Value
		<p>high number of objects. In combination with the field <code>Offset</code> this field can be used to fetch the result of a big report in multiple steps.</p> <p>If <code>Limit</code> is not set, a maximum number of 1000 result data sets of the configured report are returned.</p>
<code>"Offset":Number</code>	Optional	<p>This field defines the start position for returning values in the result data set of the configured report. For example if the <code>Limit</code> field is set to 20 and the <code>Offset</code> is set to 10, the results listed in row 11 to 30 in the tabular output of the configured report are returned in the service call. By default, the start position is the first row in the data set with the number 0.</p>
<pre>"ReportArgs": { "arg1": "val1", "arg2": "val2", }</pre>	Optional	<p>If the configured report contains filter definitions, the values to be set for the filters when executing the report via the service call must be set with this field. The field value is a JSON object with one field for each filter parameter of the configured report. The field name must be identical to the parameter name without the prefix <code>@</code> or: and the field value must be the value that shall substitute the parameter in the query.</p> <p>Please note the following:</p> <ul style="list-style-type: none"> • Values can be of the type integer, boolean or string. String values must be written in single quotes within the double quotes of the JSON (<code>"arg1": "TestString"</code>), while integers and boolean values do not require setting of single quotes. • The wildcards <code>%</code> and <code>*</code> can be used in string values. • All filter definitions of the configured report must be provided via the <code>ReportArgs</code> in the service call. If you do not want to set one of the filter values, you can define a wildcard only as value to return all results.

Field	Mandatory/Optional	Required Value
"EmptyValues":true/false	Optional	If <code>EmptyValues</code> is set to <code>true</code> , all object class properties of the object are returned even if they are not set. If <code>EmptyValues</code> is set to <code>false</code> , only object class properties of the object that are set are returned. By default the attribute is set to <code>false</code> .
"CurrentProfile": " UserProfileName "	Mandatory	The name of a user profile assigned to the Alfabet user sending the call that shall be used to access Alfabet. The user profile is required to evaluate whether the user that is sending the service call is allowed to read data about an object. For details about the access permissions that depend on the user profile, see the section Authorization .
"CurrentMandate": " MandateName "	Optional	This field is only relevant if access to objects is controlled by the mandate concept for federated architectures implemented in Alfabet. The field must specify the name of a mandate assigned to the Alfabet user sending the call. For details about the consideration of mandate settings in REST API service calls, see the section Controlling Access Via Mandates .
"Language": " LanguageCode "	Optional	<p>This field is only relevant if instance translation is used in your Alfabet application and if the service call shall return all properties of the objects found in the configured report, that means <code>ReportResult</code> is set to <code>Objects</code>. The field must specify the language code (LCID) decimal of the culture for that translations are available for the name and description of the object in the Alfabet database. By default the standard language of the database is returned.</p> <p>Please note that no values are returned when an instance translation is not available for an object class property, even if a value is provided in the standard language.</p> <p>To return all values in the original values and add translations in a second language, the parameter <code>DataCulture</code> shall be used instead of the parameter <code>Language</code>. <code>Language</code> supersedes <code>DataCulture</code>. If both fields are available, only</p>

Field	Mandatory/Optional	Required Value
		Language is used.
"ApiCulture" : "APICultureName"	Optional	<p>This field is only relevant if and if the service call shall return all properties of the objects found in the configured report, that means <code>ReportResult</code> is set to <code>Objects</code>. It specifies an API Culture that was defined in Alfabet Expand as part of the configuration of the Alfabet meta-model. The API Culture allows to define data, date/time and time patterns as well as number formats that deviate from the Alfabet standard. If this field is added to the request, dates, times and numbers are written into the return value in the formats specified in the API Culture definition. For more information, see the section Configuring Handling of Date, Time and Number Formats For API Calls.</p> <p>Please note however, that values defined in the <code>ReportArgs</code> field must be defined in Alfabet default formats independent from the API Culture.</p>

Return Value for the ReportResultDataSet Returning the Data Defined in the Report

The return value is a JSON object with four fields:

- **Count**: The number of rows in the data set of the configured report the service call is based on.
- **Name**: The name of the configured report the data is derived from as defined in the attribute **Name** of the configured report.
- **Description**: The description for the configured report the data is derived from as defined in the attribute **Description** of the configured report.
- **Objects**: A JSON list of objects, each object representing a row in the data set of the configured report. Information about the base object of the row and the values in the row of the report are given.

 {
 "Objects": [...]

```

    "Count": 3,
    "Name": "ReportName",
    "Description": "Report Description"
  }

```

The field `Objects` contains one JSON object per row in the data set of the configured report. The object contains the following fields:

- `ClassName`: The name of the Alfabet object class the base object of the current row belongs to.
- `RefStr`: The value of the `REFSTR` of the base object of the current row.
- `Values`: A JSON object that includes all information defined in the cells of the current row of the configured report. Each column in the dataset is a field with a name corresponding to the column name. The value of the field informs about the value in the current row. All values are returned as strings.

Return Value for the ReportResultObjects Returning All Properties of the Objects Found in the Report

The return value lists all object class properties set for all base objects of the configured report.

The return value is a JSON object with four fields:

- `Count`: The number of rows in the data set of the configured report the service call is based on. The count may differ from the number of objects for that data is returned. The report might display multiple rows for the same base object. In that case each base object is only considered once.
- `Name`: The name of the configured report the data is derived from as defined in the attribute **Name** of the configured report.
- `Description`: The description for the configured report the data is derived from as defined in the attribute **Description** of the configured report.
- `Objects`: A JSON list of objects, each object representing a database object for that data is returned.



```

{
  "Objects": [...],
  "Count": 3
}

```

The field `Objects` contains one JSON object per database object found for the call. The object contains the following fields:

- **ClassName:** The name of the Alfabet object class the object belongs to.
- **RefStr:**The value of the REFSTR of the object.
- **Values:** A JSON object that includes all information defined about the object in the database via object class properties. Each property is a field with a name corresponding to the property name. The value of the field informs about the value of the property. The return value for a property depends on the value type of the property:

Property Type	Return value
String, Text	The string, text is returned as string (in inverted commas): <pre>"description": "This application manages customer relations."</pre>
StringArray	All selected options of the string array are returned in one string: <pre>"subcategories": "APP_SecurityAssessment APP_CloseSubworkflow"</pre>
Boolean	Boolean values are 1 for true, 0 for false or null if the property is not set: <pre>"variant": 1</pre>
Real, Integer	Real and integer values are returned as number: <pre>"xpos": 485.4</pre>
Date, DateTime	Date and date time information is both returned as date strings of the format yyyy-mm-dd: <pre>"last_update": "2012-02-21"</pre>
URL	URLs are returned as string (in inverted commas): <pre>"url": "http://company.com/ReportServer"</pre>

Property Type	Return value
Reference	<p>The REFSTR of the object the reference is targeting is returned:</p> <pre>"ictobject": "26-608-0"</pre>
ReferenceArray	<p>For reference arrays, the REFSTRs of all objects that the current object references with this property are listed in an array:</p> <pre>"applicationgroups": [3] 0: "95-38-0" 1: "95-8-0" 2: "95-9-0"</pre> <p>The return value is independent from the setting of the attribute RefSupport of the property. If the attribute RefSupport of the property is set to <code>false</code>, the REFSTR value of the reference targets are directly stored in the database table of the object class in a column for the property. If the attribute RefSupport is set to <code>true</code>, the references are stored in the RELATIONS table. This difference is not visible in the return value for the <code>objects</code> endpoint.</p> <p>If information about the objects referenced by the current object shall be included into the return value, you can use the method to find objects via a configured report that is also available for the endpoint <code>objects</code>.</p>

- **GenericAttributes:** If attributes are defined for the object via the object class `GenericAttribute`, all generic attributes with the object class property `Owner` set to the current object are listed in this field as a JSON list of objects. Each object in the list returns one object of the class `GenericAttribute`, with the fields `Refstr`, `Values` and `Translations` that return data about the generic attribute in the same format as returned for the main object.

Return Value for the ReportResult Returning the REFSTR of the Objects Found in the Report

The return value includes all REFSTR values for the base object of each row in the configured report.

The return value is a JSON object with four fields:

- `Count:` The number of rows in the data set of the configured report the service call is based on.

- **Name:** The name of the configured report the data is derived from as defined in the attribute **Name** of the configured report.
- **Description:** The description for the configured report the data is derived from as defined in the attribute **Description** of the configured report.
- **Refs:** A JSON array containing the `REFSTR` values of the base objects in the report. The array can contain a `REFSTR` value multiple times if the base object of multiple rows is identical.

```

{
  "Refs": [ "95-36-0", "95-43-0", "95-41-0" ]
  "Count": 3
}

```

Deleting Objects from the Alfabet Database

The endpoint `delete` provide a means to delete objects and relations from the Alfabet database. A relation is a reference from one object to another object that is stored in a property of the type Reference Array. Most reference arrays are stored in the relations table of the Alfabet database.



For detailed information about the storage of relations in the Alfabet database see the chapter *The Alfabet Meta-Model in the Alfabet Database* in the reference manual *Alfabet Data Integration Framework*.

The request must include the definition of the database manipulation in the body of the request in JSON format.

Endpoint name: `delete`

HTTP method: `POST`

Service call:

```
ServerAdress/api/v2/delete
```

Header Fields:

```
Authorization:TypeValue
```



For information about the required Authorization header content, see [Authorization](#).

```
Content-Type: application/json; charset=utf-8
```

Payload

The payload is a JSON object with the following structure:

```
{
  "CurrentProfile": "UserProfile1",
  "CurrentMandate": "Mandate1",
  "Refs": ["76-2518-0", "76-3246-0", "405-3-0"],
  "Relations": [{
    "FromRef": "76-2518-0",
    "Property": "BelongsTo",
    "ToRef": "76-3246-0"
  },
  {...}]
}
```

It may have following fields:

Field	Mandatory/Optional	Required Value
"Refs":[" RefstrOfObject "," RefstrOfObject "]	Optional	An array containing one or multiple values of the REFSTR property of objects in the Alfabet database that shall be deleted.
"Relations": [{ "FromRef": "RefstrOfObject",	Optional	A JSON list of objects, each objects representing a relation to be deleted.

Field	Mandatory/Optional	Required Value
<pre>"Property": "PropertyName", "ToRef": "RefstrOfObject" }, {...}]</pre>		<p>For each relation that shall be deleted, the following fields must be defined:</p> <ul style="list-style-type: none"> <code>FromRef</code>: The REFSTR of the object for that the relation is defined via one of its object class properties. <code>Property</code>: The value of the Name attribute of the object class property for that the relation is stored. <code>ToRef</code>: The REFSTR of the object to that the relation is built.
"CurrentProfile": " UserProfileName "	Mandatory	The name of a user profile assigned to the Alfabet user sending the call that shall be used to access Alfabet. The user profile is required to evaluate whether the user that is sending the service call has write permissions to an object. For details about the access permissions that depend on the user profile, see the section Authorization .
"CurrentMandate": "MandateName"	Optional	This field is only relevant if access to objects is controlled by the mandate concept for federated architectures implemented in Alfabet. The field must specify the name of a mandate assigned to the Alfabet user sending the call. For details about the consideration of mandate settings in REST API service calls, see the section Controlling Access Via Mandates .

Return value:

The return value is a JSON object with one field "Count" that informs about the number of objects and relations that were deleted.

Creating and Updating Object Data in the Alfabet Database

The endpoint `update` provide a means to alter the content of the Alfabet database. New objects can be created for Alfabet object classes, properties of existing objects can be changed, and relations between objects can be created. The request must include the definition of the database manipulation in the body of the request in JSON format.

As a result of the service call, the content of the Alfabet database is changed and a return value is sent that confirms the change.

Endpoint name: `update`

HTTP method: `PUT`

Service call:

```
ServerAdress/api/v2/update
```

Header Fields:

```
Authorization:TypeValue
```



For information about the required Authorization header content, see [Authorization](#).

```
Content-Type: application/json;charset=utf-8
```

Payload

The payload is a JSON object with the following structure:

```
{
  "CurrentProfile": "UserProfile1",
  "CurrentMandate": "Mandat1",
  "ApiCulture": "ApiCultureName",
  "Objects": [...],
  "Relations": [...],
}
```

It may have following fields:

Field	Mandatory/Optional	Required Value
"Objects"	Optional	An array containing the specification of the objects and the data that shall be changed for the objects. Details are described below in the sections Creating a new Object in the Alfabet database and Changing the Properties of an existing Object in the Alfabet database .
"Relations"	Optional	An array containing the specification of the references that shall be changed. Details are described below in the section Creating or Updating a Relation Between Objects in the Alfabet database .
"CurrentProfile"	Mandatory	The name of a user profile assigned to the Alfabet user sending the call that shall be used to access Alfabet. The user profile is required to evaluate whether the user that is sending the service call has write permissions to an object. For details about the access permissions that depend on the user profile, see the section Authorization .
"CurrentMandate"	Optional	This field is only relevant if access to objects is controlled by the mandate concept for federated architectures implemented in Alfabet. The field must specify the name of a mandate assigned to the Alfabet user sending the call. For details about the consideration of mandate settings in REST API service calls, see the section Controlling Access Via Mandates .
"ApiCulture"	Optional	This field specifies an API Culture that was defined in Alfabet Expand as part of the configuration of the Alfabet meta-model. The API Culture allows to define data, date/time and time patterns as well as number formats that deviate from the Alfabet standard. If this field is added to the request, dates, times and numbers can be defined in the object data definitions for update of object data in the formats specified in the API Culture definition. For more information, see the section Configuring Handling of Date, Time and Number Formats For API Calls .

The following sections describe the different kind of operations and the required JSON request:

- [Creating a new Object in the Alfabet database](#)
- [Changing the Properties of an existing Object in the Alfabet database](#)

- [Creating or Updating a Relation Between Objects in the Alfabet database](#)

A single JSON request can include multiple different operations, that means you can for example create new objects and update data of existing objects in the same request.



The update of the data in the Alfabet database requires knowledge about the object class configuration of the Alfabet meta-model and/or existing object data.

- Information about the current definition of object classes and object class properties can be retrieved via the endpoint `metamodel`.
- Information about the current object data can be retrieved via the endpoint `select` or `object`.
- For basic information about the structure of the Alfabet class model that is required to perform data manipulation directly on the database level, see *The Alfabet Meta-Model in the Alfabet Database* in the reference manual *Alfabet Data Integration Framework*.

Return value:

The return value is a JSON object informing about changes performed and errors that occurred, for example when creating a two new objects, the return value is:

```
{
  "NewObjects": {
    1: "95-61-0"
    2: "95-62-0" }
  "Count": 2
}
```

The return value can have the following fields:

Field	Value
"NewObjects"	This field contains a JSON object with one field for each new object. The field name is the Id defined in the request and the field value is the value of the <code>REFSTR</code> property of the new object in the Alfabet database. If the request only updates objects, this field is empty.

Field	Value
"Access Denied"	This field contains a JSON object with a field for each object for that access is denied for example because of mandate settings. The field name is the REFSTR value of the object and the field value is the reason for denial of access.
"RejectedObjects"	This field informs about object that could not be created or changed. The field contains a JSON list of objects with one JSON object for each rejected change to an Alfabet object. The object has a field <code>RefStr</code> that returns the <code>REFSTR</code> of the Alfabet object and a field <code>Message</code> that gives information about the rejection.
"NotFound"	If a relation shall be created and one or both of the involved objects does not exist in the Alfabet database, the call displays a <code>NotFound</code> field containing an array with the <code>REFSTR</code> values that could not be found.
"Count"	This fields informs about the number of objects and relations that were updated or created.

Creating a New Object in the Alfabet Database

The JSON object of the request to create a one or multiple new objects contains the following fields:

- `CurrentProfile`: The name of the user profile that shall be used for the user when performing the call. The permission to create an object of an object class in the Alfabet database is evaluated via the user profile. Detailed information is given in the section. The field is optional. If it is not included, the last user profile the user was logged in with is used for the call. If the information about the user profile of the last log in is not found, the user profile defined as default profile for the user in the user settings of the user is used as default for the field.
- `Objects`: A definition of object data that shall be created. Details are given below.

The value of the field `Objects` is a JSON list of objects, each object defining one Alfabet object to be created with the following fields:

Field name	Required Value	Remark
ClassName	The value of the Name attribute of the Alfabet object class for that the object shall be created.	
Id	An integer that is a unique identifier inside the JSON request.	Each object created in a single JSON request must have a different id. The id is only used within the single JSON request. It is not having any impact on the Alfabet database. As soon as the objects are created, the ids are no longer coupled to the objects and can be used in other service calls.
Values	<p>A JSON object with one field for each property that shall be defined.</p> <p>The field name is identical to the Name of the object class property written in lower case letters.</p> <p>The field value is identical to the value of the object class property of the created object.</p>	<p>At least all properties defined as mandatory must be set for an object to create the object.</p> <p>The property <code>REFSTR</code> cannot be defined in the JSON request. It is set automatically during creation of the object and returned in the return value of the service call. Technical properties like creation date and creation user are also automatically set during creation of the object. Properties with the attribute Automatically Managed set to <code>true</code> must not be set via this mechanism.</p> <p>Properties of the type Reference and ReferenceArray cannot be updated with this mechanism. For information about the update of properties of the type Reference and ReferenceArray see Creating or Updating a Relation Between Objects in the Alfabet database.</p> <p>The value must match the format and restrictions that apply to the object class property. For example a string length restriction might be implemented for an object class property of the type string. Dates must be defined in the format yyyy-mm-dd.</p>

Field name	Required Value	Remark
Translations	A list of JSON objects, each object representing a translation and having two fields: the field <code>DataCulture</code> that defines the language code of the data culture (for example <code>fr-FR</code>) and the field <code>Values</code> , that contains a field for each translation to be added with the field name being identical to the property name and the field value being identical to the translation that shall be added.	This field is optional and shall be added if translations of object class names and descriptions shall be provided for one or multiple data cultures specified in Alfabet Expand that are defined to allow data translation.
GenericAttributes	<p>A JSON list of objects, with one JSON object for each generic attribute to create or update. Each JSON object has three fields:</p> <ul style="list-style-type: none"> <code>RefStr</code>: The field must be defined with an empty value to create a new generic attribute. <code>Values</code>: A JSON object with one field for each property that shall be updated. Required fields are <code>Name</code>, <code>Type</code> and <code>Value</code>. The field <code>Group</code> is optional. The property <code>Owner</code> of the object class <code>GenericAttribute</code> must not be specified. It is set by the import mechanism to the <code>REFSTR</code> to the object the generic attribute is defined for. <code>Translations</code>: If the name of the generic attribute shall be translated, a field translation can be specified as described above for the main object. 	The object class <code>GenericAttribute</code> stores properties for different object classes. If generic attributes shall be defined for objects created via an endpoint <code>update</code> , generic reference data must be created directly within the request for the object it belongs to.



The following example shows the required JSON request for creating two application groups with a translation into two languages and a generic attribute:

```
{
  "CurrentProfile": "RESTAccessProfile";
```

```
"Objects": [{
  "ClassName": "ApplicationGroup",
  "Id": "1",
  "Values": {
    "name": "Test Group 1",
    "shortname": "TestG1"
  }
  "Translations":[
    {"DataCulture": "de-DE",
     "Values":{"name": "Testgruppe 1"}
    },
    {"DataCulture": "fr-FR",
     "Values":{"name": "Groupe Controle 1"}
    }
  ]
},
{
  "ClassName": "ApplicationGroup",
  "Id": "2",
  "Values": {
    "name": "Test Group 2",
    "shortname": "TestG2"
  },
  "Translations":[
    {"DataCulture": "de-DE",
     "Values":{"name": "Testgruppe 2"}
    }
  ]
}
```

```

    },
    {"DataCulture": "fr-FR",
     "Values": {"name": "Groupe Controle 2"}
    }
  ],
  "GenericAttributes": [{
    "RefStr": "",
    "Values": {"Name": "Integer Attribute 1",
              "Group": "",
              "Type": "Integer",
              "Value": "99999"},
    "Translations": [{
      "DataCulture": "de-DE",
      "Values": {"Name": "Integer-Attribut"}
    }]
  }]
}

```

Changing the Properties of an Existing Object in the Alfabet Database

To update property values for an existing object in the Alfabet database, the JSON object of the request must contain a field `objects`. The value of this field is a JSON list of objects, each object defining one Alfabet object to be updated with the following fields:

Field name	Required Value	Remark
REFSTR	The value of the REFSTR property of the object that shall be updated.	
Values	<p>A JSON object with one field for each property that shall be updated.</p> <p>The field name is identical to the Name of the object class property written in lower case letters.</p> <p>The field value is identical to the new value that shall be set for the object class property.</p>	<p>Properties with the attribute Automatically Managed set to <code>true</code> must not be set via this mechanism.</p> <p>Properties of the type Reference and ReferenceArray cannot be updated with this mechanism. For information about the update of properties of the type Reference and ReferenceArray see Creating or Updating a Relation Between Objects in the Alfabet database.</p> <p>The value must match the format and restrictions that apply to the object class property. For example a string length restriction might be implemented for an object class property of the type string. Dates must be defined in the format yyyy-mm-dd.</p>
Translations	A list of JSON objects, each object representing a translation and having two fields: the field <code>DataCulture</code> that defines the language code of the data culture (for example <code>fr-FR</code>) and the field <code>Values</code> , that contains a field for each translation to be added with the field name being identical to the property name and the field value being identical to the translation that shall be added.	This field is optional and shall be added if translations of object class names and descriptions shall be provided for one or multiple data cultures specified in Alfabet Expand that are defined to allow data translation.
GenericAttributes	<p>A JSON list of objects, with one JSON object for each generic attribute to create or update. Each JSON object has three fields:</p> <ul style="list-style-type: none"> <code>RefStr</code>: For updates, the value of the field must be the REFSTR of the existing generic attribute. For creating a new generic attribute, the field must be defined with an empty value. 	The object class <code>GenericAttribute</code> stores properties for different object classes. If generic attributes shall be defined via an endpoint <code>update</code> , generic reference data must be updated or created directly within the update request for the object it belongs to.

Field name	Required Value	Remark
	<ul style="list-style-type: none"> • Values: A JSON object with one field for each property that shall be updated. Required fields are <code>Name</code>, <code>Type</code> and <code>Value</code>. The field <code>Group</code> is optional. The property <code>Owner</code> of the object class <code>Generic Attribute</code> must not be specified. It is set by the import mechanism to the <code>REFSTR</code> to the object the generic attribute is defined for. • Translations: If the name of the generic attribute shall be translated, a field translation can be specified as described above for the main object. 	



The following example shows the required JSON request for updating two application groups including translation of the names for the two objects:

```
{
  "Objects": [{
    "RefStr": "95-61-0",
    "Values": {
      "shortname": "TG1",
      "status": "Planned"
    }
  },
  {
    "Translations": [
      { "DataCulture": "de-DE",
        "Values": { "name": "Testgruppe 1" }
      },
      { "DataCulture": "fr-FR",
        "Values": { "name": "Groupe Controle 1" }
      }
    ]
  }
]
```

```
]
"GenericAttributes": [{
  "RefStr": "",
  "Values": {"Name": "Integer Attribute 1",
  "Group": "",
  "Type": "Integer",
  "Value": "99999"},
  "Translations": [{
    "DataCulture": "de-DE",
    "Values": {"Name": "Integer-Attribut"}
  }]
}]
},
{
"RefStr": "95-62-0",
"Values": {
  "shortname": "TG2",
}
"Translations":[
  {"DataCulture": "de-DE",
  "Values":{"name": "Testgruppe 1"}
},
  {"DataCulture": "fr-FR",
  "Values":{"name": "Groupe Controle 1"}
}
]
}]
```

}

Creating or Updating a Relation Between Objects in the Alfabet Database

Relations can be created between objects that already exist in the Alfabet database. If an object does not exist, the relation is not set and the REFSTR of the object or objects that could not be found is returned in an array in a JSON field `NotFound` in the return value.

A relation between two objects is established via a property of one of the objects that is of the type `Reference` or `ReferenceArray`. Although different mechanisms exist for storage of relations for properties of the type `Reference` and `ReferenceArray`, these differences are of no importance for the definition of the relation in the service call. This is handled by the Alfabet RESTful API on server side.

If a relation is defined that already exists in the Alfabet database, the following happens:

- If the object class property storing the relation is of the **Type** `Reference`, the existing reference is overwritten.
- If the object class property storing the relation is of the **Type** `ReferenceArray`, the new reference is added to the array and existing references persist. If you want a relation of the **Type** `ReferenceArray` to be substituted, you must delete the old relation with a service request of the endpoint `delete` and create a new one with this service request.

For the request send with the service call, the following information must be provided:

- Definition of the "from" object. This is the object for that the property establishing the relation is defined.
- Definition of the "to" object. This is the object to that the relation is established.
- Definition of the property establishing the relation.

To create a new relation, the JSON object of the request must contain a field `Relations`. The value of this field is a JSON list of objects, each object defining one relation between Alfabet objects to be created or updated with the following fields:

Field name	Required Value
<code>FromRef</code>	The value of the <code>REFSTR</code> property of the Alfabet object for that the property establishing the relation is defined.

Field name	Required Value
Property	The value of the Name attribute of the object class property establishing the relation.
ToRef	The value of the <code>REFSTR</code> property of the Alfabet object to that the relation is established.



The following example shows a JSON request that updates a property of the type Reference and creates a relation for a property of the type ReferenceArray. The updated object is an application group. The property `ResponsibleUser` is of the type Reference. The existing reference to a responsible user is therefore overwritten with the relation defined in the request. The property `Applications` is of the type ReferenceArray. The relation defined in the request is added to already existing relations:

```
{
  "Relations": [{
    "FromRef": "95-61-0",
    "Property": "ResponsibleUser",
    "ToRef": "421-862-0"
  },
  {
    "FromRef": "95-61-0",
    "Property": "Applications",
    "ToRef": "76-2518-0"
  }
  ]
}
```

Archiving Objects from the Alfabet Database

The endpoint `archiveobject` provide a means to create an archive ZIP file for one or multiple selected objects from the Alfabet database. The object can optionally be deleted after creating the archive ZIP file.

When an Alfabet object is archived, a ZIP file is created containing HTML files that display the object profile for the archived object as well as the object profiles of its dependent objects. Each archived object profile displays a preconfigured set of page views, whereby the visibility of these views will depend on the class setting configured for the object class. If a page view displays dependent objects, a user can click the dependent object in the HTML view in order to open another HTML file showing the archived object profile of the selected dependent object.

Alfabet objects are typically archived by a solution administrator in the **Simple Search** functionality of the **Admin** user profile. If an archive is created via the Alfabet user interface, the archived Alfabet object is deleted from the Alfabet database after generation of the archive ZIP file. In the **Archive Manager** functionality of the **Admin** user profile, the archive ZIP file is then available for download to a local disk. After extracting the ZIP file, the relevant HTML file can then be viewed in a browser window. The archive ZIP file contains one folder for each culture setting supported by your enterprise.

This functionality is especially useful in order to remove outdated objects from the Alfabet database and thus enhance the performance of Alfabet. The archived data can be saved and displayed when needed in a Web browser.



For information about the archiving of Alfabet objects via the Alfabet user interface, see *Deleting and Archiving Alfabet Objects* in the reference manual *User and Solution Administration*.

Archiving of objects via a RESTful service call provides more flexibility:

- An archive can be created without deleting the object.
- Archives can either be made available in the **Archive Manager** functionality of the Alfabet user interface only, or they can additionally be directly stored in a defined folder on the local file system directly during the execution of the RESTful service call.

The request must include the definition of the archiving and database manipulation in the body of the request in JSON format.

Endpoint name: `archiveobject`

HTTP method: `POST`

Service call:

`ServerAddress/api/v2/archiveobject`

Header Fields:

Authorization: *TypeValue*



For information about the required Authorization header content, see [Authorization](#).

Content-Type: application/json; charset=utf-8

Payload

The payload is a JSON object with the following structure:

```
{
  "CurrentProfile": "UserProfile1",
  "CurrentMandate": "Mandate1",
  "CurrentCulture": "1033",
  "Refs": ["76-2518-0", "76-3246-0", "405-3-0"],
  "DeleteAfterArchive": "true",
  "ZipPath": "C:\\\\ArchiveObject\\\\Applications"
}
```

It may have following fields:

Field	Mandatory/Optional	Required Value
"Refs":[" RefstrOfObject ", " RefstrOfObject "]	Mandatory	An array containing one or multiple values of the REFSTR property of objects in the Alfabet database that shall be archived or archived and deleted.
"CurrentProfile": " UserProfileName "	Mandatory	The name of a user profile assigned to the Alfabet user sending the call that shall be used to access Alfabet. The user profile is required to evaluate whether the user that is sending the service call has write permissions to an

Field	Mandatory/Optional	Required Value
		object. For details about the access permissions that depend on the user profile, see the section Authorization .
"CurrentMandate": "MandateName "	Optional	This field is only relevant if access to objects is controlled by the mandate concept for federated architectures implemented in Alfabet. The field must specify the name of a mandate assigned to the Alfabet user sending the call. For details about the consideration of mandate settings in REST API service calls, see the section Controlling Access Via Mandates .
"CurrentCulture": "LCID decimal "	Optional	The language code (LCID) decimal of the culture that shall be used for messages in the return value. The culture must be a culture for that translations are available via the vocabulary files in the Alfabet database. By default, the message in the return value are English.
"ZipPath": "AbsolutePathToFolder "	Optional	<p>The absolute path to the folder on the local file system in which the archive should be stored in addition to being stored in the Alfabet database. If the folder does not exist, it is created during execution of the service call.</p> <p>If this field is not set, the archive is only stored in the Alfabet database and available for download via the Archive Manager functionality on the Alfabet user interface.</p> <p>Please note that the back slashes in the path must be written as double back slashes to be accepted as text in the JSON definition.</p>
"DeleteAfterArchive": "true false"	Optional	<p>Set the field value to <code>true</code> if the object shall be deleted after generation of the archive. By default, the object is not deleted after generation of the archive.</p> <p>Additional access rights are required for deletion of the object. If you set the</p>

Field	Mandatory/Optional	Required Value
		field to <code>true</code> and the permissions for the user and user profile for processing the call are not sufficient to delete the object, the archive is also not created, even if the permissions for creating archives are given.

Return value:

The return value is a JSON object with one field "ResultMessages" that informs about the success status in a separate field per object in the format "REFSTR of object": "Message".

```
{
  "ResultMessages": {
    "95-44-0": "Instance not found",
    "76-3200-0": "Access denied: 'Instance cannot be deleted'",
    "95-43-0": "Object successfully archived."
  }
}
```

Regenerating the Password of an Alfabet User

The endpoint `regeneratepassword` provide a means to reset the password of a user or to create an initial password for a new user. A password is automatically assigned to the users and the user receives emails informing the him/her about the login credentials.



This functionality includes sending emails to the user via the system. You must ensure that system emails are activated for your Alfabet installation. For more information about activating the sending of emails, see the section *Activating the Dispatch of Email Notifications in Alfabet* in the reference manual *System Administration*. For more information about specifying the message in the emails or configuring custom text templates to use in place of the standard text templates, see the section *Specifying Custom Text Templates for Password Generation* in the reference manual *Configuring Alfabet with Alfabet Expand* and the section *Text Templates for Activation of User Passwords* in the reference manual *Configuring Alfabet with Alfabet Expand*

- Appendix.

Password regeneration actions are logged in a log file. This is by default the file `PasswordChangeLog.txt` in the working directory of the Alfabet Web Application. For information about how to change the location of the log file, see *Defining the Location of the Log File for Password Reset and Regeneration Actions* in the reference manual.



For general information about the configuration and administration requirements for user login via user name and password, see *Configuring Standard Login* in the reference manual *System Administration*.

The request must include the definition of the archiving and database manipulation in the body of the request in JSON format.

Endpoint name: `regeneratepassword`

HTTP method: `POST`

Service call:

```
ServerAddress/api/v2/regeneratepassword
```

Header Fields:

```
Authorization:TypeValue
```



For information about the required Authorization header content, see [Authorization](#).

```
Content-Type: application/json; charset=utf-8
```

Payload

The payload is a JSON object with the following structure:

```
{
  "CurrentProfile": "UserProfile1",
  "CurrentCulture": "1033",
  "Refs": ["421-2518-0", "421-3-0"],
```

```
"UserNames": [ "CUSTOMER" , "CLIENTE" ]
}
```

It may have following fields:

Field	Mandatory/Optional	Required Value
"Refs":[" RefstrOfObject " ," RefstrOfObject "]	Optional	An array containing one or multiple values of the REFSTR property of the users (object class <code>Person</code>) in the Alfabet database for which the password should be regenerated.
"UserNames": ["CUSTOMER" , "CLIENTE"]	Optional	An array containing one or multiple user names of users (<code>USER_NAME</code> property of the object class <code>Person</code>) in the Alfabet database for which the password should be regenerated.
"CurrentProfile": " UserProfileName "	Mandatory	The name of a user profile assigned to the Alfabet user sending the call that shall be used to access Alfabet. The user profile is required to evaluate whether the user that is sending the service call has write permissions to an object. For details about the access permissions that depend on the user profile, see the section Authorization .
"CurrentCulture": " LCID decimal "	Optional	The language code (LCID) decimal of the culture that shall be used for messages in the return value. The culture must be a culture for that translations are available via the vocabulary files in the Alfabet database. By default, the message in the return value are English.

Return value:

The return value is a JSON object with one field `"ResultMessages"` that informs about the success status in a separate field per defined user in the format `"REFSTR of user": "Message"` or `"User Name": "Message"`.

```
{
  "ResultMessages": {
    "421-2518-0": "User not found",
    "421-3-0": "Password successfully changed.",
    "CUSTOMER": "Password successfully changed.",
    "CLIENTE": "Password successfully changed."
  }
}
```

Anonymizing User Data for Selected Users

This endpoint can be used to anonymize the data of one or multiple selected Alfabet users. The endpoint will only anonymize the data if the following preconditions are met:

- Anonymization is enabled in the Alfabet database for the object class `PERSON`. For more information about the anonymization feature and the required configuration to enable it, see *Anonymizing Data* in the reference manual *Configuring Alfabet with Alfabet Expand*.
- The user is not configured to be excluded from anonymization. For more information about excluding users from anonymization, see *Excluding Users from Anonymization* in the reference manual *Configuring Alfabet with Alfabet Expand*.
- The Alfabet user used to execute the call has the required access permission **Has AnonymizeUser Access**. For more information about access permissions for REST API service calls, see [Generating an API Password for a User for General Use of the Alfabet RESTful API](#).



Data anonymization is an irreversible action!

Endpoint name: `anonymizeuser`

HTTP method: `PUT`

Service call:

`ServerAddress/api/v2/anonymizeuser`

Header Fields:

`Authorization:TypeValue`



For information about the required Authorization header content, see [Authorization](#).

`Content-Type: application/json; charset=utf-8`

Payload

The payload is a JSON object with the following structure:

```
{
  "Refs": ["421-5-0", "421-9-0"]
}
```

It may have the following fields:

Field	Mandatory/Optional	Required Value
"Refs":[" RefstrOfObject "," RefstrOfObject "]	Mandatory	An array containing one or multiple values of the REFSTR property of objects of the object class <code>Person</code> with the stereotype <code>User</code> in the Alfabet database. All users included into the array will be anonymized.

Return value:

The return value is a JSON object with three fields:

- `Count`: The number of users that have been anonymized.
- `RejectedObjects`: A JSON list of objects that were not anonymized, for example because the user is excluded from anonymization. The field contains a JSON list of objects with one JSON object for each rejected Alfabet object. The object has a field `RefStr` that returns the `REFSTR` of the Alfabet object and a field `Message` that gives information about the rejection.

```
 {  
  "Count": 3,  
  "RejectedObjects": {  
    "RefStr": "421-999-0",  
    "Message": "Cannot find instance"  
  }  
}
```

Starting a Workflow via RESTful Service Call

This endpoint can be used to start a workflow that is configured to be automatically started and to allow start via REST API.

Endpoint name: `workflow`

HTTP method: `POST`

Service call:

`ServerAddress/api/v2/workflow`

Header Fields:

`Authorization:TypeValue`



For information about the required Authorization header content, see [Authorization](#).

`Content-Type: application/json; charset=utf-8`

Payload

The payload is a JSON object with the following structure:

```
{
  "WorkflowTemplate": "WorkflowTemplateName"
}
```

It may have the following fields:

Field	Mandatory/Optional	Required Value
"WorkflowTemplate": " WorkflowTemplateName "	Mandatory	The name of the workflow template that shall be started.

Return value:

The return value is a JSON object that can show the following fields:

- **ResultMessage:** A message informing about the number of workflows that have been started.
- **ErrorMessage:** If starting of the workflow fails, this field informs about the reason for that.
- **ErrorCode:** If starting of the workflow fails, this field returns the error code for the error. This information is only relevant if you can not fix the problem because of the information in the error message. You can then send the error code and error message to your system administrator for support.



```
{
  "ResultMessage": "3 new workflows have been created."
}
```

Starting an ADIF Import via RESTful Service Call

This endpoint can be used to start an ADIF import based on an ADIF import scheme stored in the Alfabet database targeted by the REST API call. ADIF import schemes can only be started if the following applies to the ADIF scheme:

- The attribute **Applicable for REST API** is set to `True`.

- No files on the local file system are involved, which means that only ADIF import schemes importing data from another database can be executed via REST API.

Endpoint name: `adifimport`

HTTP method: `POST`

Service call:

`ServerAddress/api/v2/adifimport`

Header Fields:

`Authorization:TypeValue`



For information about the required Authorization header content, see [Authorization](#).

`Content-Type: application/json; charset=utf-8`

Payload

The payload is a JSON object with the following structure:

```
{
  "Scheme": "ADIFImportSchemeName",
  "UserArgs": { "@AppRef": "76-2760-0" },
  "Verbose": false,
  "Synchron": false
}
```

It may have the following fields:

Field	Mandatory/Optional	Required Value
"Scheme": " ADIFImportSchemeName "	Mandatory	The name of the ADIF import scheme that shall be executed.
"UserArgs": { " arg1name ":" arg1value "," arg2name ":" " arg2value " }	Optional, only required if the ADIF scheme uses variables.	When the ADIF scheme is configured to use variables, the variables can be specified in the field <code>UserArgs</code> as a JSON object with one field for each variable. The field name must be identical to the variable name and the field value defines the variable value for the current execution of the ADIF import scheme.
"Verbose": true/false	Optional, default is false	If this field is set to <code>true</code> , additional information about the running process is logged in the logging information in the <code>ALFA_ADIF_SESSION</code> table. NOTE: Verbose logging is in most cases not required and can lead to a decrease in performance.
"Synchron": true/false	Optional, default is true	Specifies whether the process shall wait for the ADIF job to be executed. This field is only evaluated if the Alfabet Web Application is configured to execute ADIF jobs via an Alfabet Server. Direct execution of the ADIF job via the Alfabet Web Application is always synchronous. If <code>Synchron</code> is set to <code>false</code> , the REST API call is completed even if the ADIF job is still scheduled or processed. If <code>Synchron</code> is set to <code>true</code> , the REST API call is only completed after the ADIF job has been executed and returns overall success of the REST API call and the ADIF job. Please note that the return value of the ADIF job informs about the success of the service call only. No feedback about the success of the ADIF job execution is provided via the service call.

Return value:

The return value is a JSON object that informs about the success of the REST API service call execution. It does not provide information about the execution of the ADIF job.

Please note that trying to start an ADIF export scheme with the `adifimport` endpoint will fail. The error message does not inform about the wrong type of scheme but states that the ADIF scheme is not found. This is due to the fact that the REST API searches for the ADIF scheme name in the subset of ADIF import schemes only and therefore the ADIF export scheme is not found.

Starting an ADIF Export via RESTful Service Call

This endpoint can be used to start an ADIF export based on an ADIF export scheme stored in the Alfabet database targeted by the REST API call. ADIF export schemes can only be started if attribute **Applicable for REST API** is set to `True`.

If the ADIF export scheme is configured to export to file, the generated file or files are added to a ZIP file that is send back as a stream.

Optionally, you can decide to upload the created stream back into a file in the Internal Document Selector in the Alfabet database. The exported content will then not be available via a stream via the return value but directly written into the file in the Internal Document Selector.



For information about the Internal Document Selector, see *Uploading Documents and Managing User Permissions to Document Folders in the Internal Document Selector* in the reference manual *User and Solution Administration* and *Making Documents and Files Available to the Alfabet User Community* in the reference manual *System Administration*.

Endpoint name: `adifexport`

HTTP method: `POST`

Service call:

```
ServerAddress/api/v2/adifexport
```

Header Fields:

```
Authorization:TypeValue
```



For information about the required Authorization header content, see [Authorization](#).

```
Content-Type: application/json; charset=utf-8
```

Payload

The payload is a JSON object with the following structure:

```
{
  "Scheme": "ADIFExportSchemeName",
  "UserArgs": { "@AppRef": "76-2760-0" },
  "Verbose": false,
  "Synchron": false
}
```

It may have the following fields:

Field	Mandatory/Optional	Required Value
"Scheme": " ADIFExportSchemeName "	Mandatory	The name of the ADIF export scheme that shall be executed.
"UserArgs": { " arg1name ":" arg1value "," arg2name ":" " arg2value " }	Optional, only required if the ADIF scheme uses variables.	When the ADIF scheme is configured to use variables, the variables can be specified in the field <code>UserArgs</code> as a JSON object with one field for each variable. The field name must be identical to the variable name and the field value defines the variable value for the current execution of the ADIF export scheme.
"Verbose": true/false	Optional, default is <code>false</code>	If this field is set to <code>true</code> , additional information about the running process is logged in the logging information in the <code>AL-FA_ADIF_SESSION</code> table. NOTE: Verbose logging is in most cases not required and can lead to a decrease in performance.

Field	Mandatory/Optional	Required Value
"Synchron": true/false	Optional, default is false	<p>Specifies whether the process shall wait for the ADIF job to be executed. This field is only evaluated if the Alfabet Web Application is configured to execute ADIF jobs via an Alfabet Server. Direct execution of the ADIF job via the Alfabet Web Application is always synchronous. If <code>Synchron</code> is set to <code>false</code>, the REST API call is completed even if the ADIF job is still scheduled or processed. If <code>Synchron</code> is set to <code>true</code>, the REST API call is only completed after the ADIF job has been executed.</p> <p>ADIf export to file must be started synchronously.</p> <p>Please note that the return value of the ADIF job informs about the success of the service call only. No feedback about the success of the ADIF job execution is provided via the service call.</p>
"FolderPath":"IDOC:\\FolderName"	Optional, but mandatory for upload of exported results to the Internal Document Selector.	<p>If this field and the <code>ExportFileName</code> field are defined correctly and the executed ADIF scheme is exporting to file, the export result will be stored in the Internal Document Selector in the folder defined with this field with the file name defined in the <code>ExportFileName</code> field.</p> <p>The path must start with <code>IDOC:\</code> and backslashes must be used between folder names. Please note that backslashes must be escaped with another backslash in JSON. If you are including the JSON into code when implementing a RESTful client, further escaping of the backslashes required by JSON may be required for code execution. You might require setting four backslashes between folder names.</p> <p>It is not possible to upload documents directly into the <code>IDOC:\</code> root.</p>

Field	Mandatory/Optional	Required Value
"ExportFileName": "Filename.zip"	Optional, but mandatory for upload of exported results to the Internal Document Selector.	<p>If this field and the <code>ExportFileName</code> field are defined correctly and the executed ADIF scheme is exporting to file, the export result will be stored in the Internal Document Selector in the folder defined with this field with the file name defined in the <code>ExportFileName</code> field.</p> <p>If the file already exists in the Internal Document Selector, it will be overwritten.</p>

Return value:

For successful ADIF export to file, the return value is a stream containing the ZIP file with the exported information.

Otherwise the return value is a JSON object that informs about the success of the REST API service call execution. It does not provide information about the execution of the ADIF job.

Please note that trying to start an ADIF import scheme with the `adifexport` endpoint will fail. The error message does not inform about the wrong type of scheme but states that the ADIF scheme is not found. This is due to the fact that the REST API searches for the ADIF scheme name in the subset of ADIF export schemes only and therefore the ADIF import scheme is not found.

Exporting Information about the Content of the Internal Document Selector

This endpoint can be used to export information about the content of the Internal Document Selector of the Alfabet database or one of its sub-folders.



For information about the Internal Document Selector, see *Uploading Documents and Managing User Permissions to Document Folders in the Internal Document Selector* in the reference manual *User and Solution Administration* and *Making Documents and Files Available to the Alfabet User Community* in the reference manual *System Administration*.

Endpoint name: `idocfilelist`

HTTP method: POST

Service call:

```
ServerAdress/api/v2/idocfilelist
```

Header Fields:

```
Authorization:TypeValue
```



For information about the required Authorization header content, see [Authorization](#).

```
Content-Type: application/json; charset=utf-8
```

Payload:

The payload is a JSON object with the following structure:

```
{
  "Path": "IDOC:\\Folder\\Subfolder"
}
```

It may have the following fields:

Field	Mandatory/Optional	Required Value
"Path": " Path to folder in IDOC "	Mandatory	<p>The path to the IDOC folder the list shall be created for. The path must start with IDOC:\ and backslashes must be used between folder names.</p> <p>Please note that backslashes must be escaped with another backslash in JSON.</p> <p>If you are including the JSON into code when implementing a RESTful client, further escaping of the backslashes required by JSON may be required for code execution. You might require setting four backslashes between folder names.</p>

Return value:

The return value is a JSON list of objects with one object per document that informs about the document with the following fields:

- **Path:** The complete path of the document starting from the root of the Internal Document Selector.
- **Name:** The name of the document.
- **CreationDate:** The date and time the document was uploaded into the Internal Document Selector.
- **DateModified:** The date and time the document was last changed in the Internal Document Selector.
- **Size:** The size of the document in byte.



```
{
  "Files": [
    {
      "Path": "IDOC:\\Documents\\CC_Corporate_FI-CO.pdf",
      "Name": "CC_Corporate_FI-CO.pdf",
      "CreationDate": "2008-10-03T13:45:48.5",
      "DateModified": "2010-04-04T13:45:48",
      "Size": 24117
    },
    {
      "Path": "IDOC:\\Documents\\CC_TradeWeb.pdf",
      "Name": "CC_TradeWeb.pdf",
      "CreationDate": "2007-08-30T13:46:09.14",
      "DateModified": "2009-02-28T13:46:09",
      "Size": 24892
    }
  ]
}
```

Downloading Documents from the Internal Document Selector

This endpoint can be used to download files from the Internal Document Selector of the Alfabet database or one of its sub-folders.



For information about the Internal Document Selector, see *Uploading Documents and Managing User Permissions to Document Folders in the Internal Document Selector* in the reference manual *User and Solution Administration* and *Making Documents and Files Available to the Alfabet User Community* in the reference manual *System Administration*.

Endpoint name: `idocdownload`

HTTP method: `POST`

Service call:

```
ServerAdress/api/v2/idocdownload
```

Header Fields:

```
Authorization:TypeValue
```



For information about the required Authorization header content, see [Authorization](#).

```
Content-Type: application/json; charset=utf-8
```

Payload:

The payload is a JSON object with the following structure:

```
{
  "Path": "IDOC:\\Folder\\Subfolder\\FileName.FileExtension"
}
```

It may have the following fields:

Field	Mandatory/Optional	Required Value
"Path": " Path to file in IDOC "	Mandatory	<p>The path to the IDOC file that shall be downloaded including the file name. The path must start with IDOC:\ and backslashes must be used between folder names.</p> <p>Please note that backslashes must be escaped with another backslash in JSON.</p> <p>If you are including the JSON into code when implementing a RESTful client, further escaping of the backslashes required by JSON may be required for code execution. You might require setting four backslashes between folder names.</p>

Return value:

The return value of the successful service call provides the downloaded file in a stream.

Uploading Documents to the Internal Document Selector

This endpoint can be used to export information about the content of the Internal Document Selector of the Alfabet database or one of its sub-folders.



For information about the Internal Document Selector, see *Uploading Documents and Managing User Permissions to Document Folders in the Internal Document Selector* in the reference manual *User and Solution Administration* and *Making Documents and Files Available to the Alfabet User Community* in the reference manual *System Administration*.



Upload of information into the Internal Document Selector can also be performed directly for ADIF export to file executed via the RESTful service endpoint `adifexport`. The exported content is then not provided for download, but directly stored in the Internal Document Selector. For more information, see [Starting an ADIF Export via RESTful Service Call](#).

Endpoint name: `idocupload`

HTTP method: `POST`

Service call:

```
ServerAdress/api/v2/idocupload
```

Header Fields:

```
Authorization:TypeValue
```



For information about the required Authorization header content, see [Authorization](#).

```
Content-Type: multipart/form-data; charset=utf-8; delimiter:DelimiterStringNotPartOfAnyContent
```

Payload:

The payload consists of two parts. The payload needs to start and end with the delimiter defined in the Content-Type header field, and the delimiter must also be used to separate the two parts of the payload. The two parts of the payload must be the following, defined in the given order:

- 1) A JSON object providing details about the storage of the file in the Internal Document Selector.
- 2) The content of the file as a stream.

```
---somedelimiter--unique--
{
  "FolderPath": "IDOC:\\Folder",
  "Name": "FileName.FileExtention"
  "OverwriteExistingFile": "true"
}
---somedelimiter--unique--
file content stream data
---somedelimiter--unique--
```

The JSON may have the following fields:

Field	Mandatory/Optional	Required Value
"FolderPath": " Path to folder in IDOC "	Mandatory	<p>The path to the IDOC folder the file should be uploaded to. The path must start with IDOC:\ and backslashes must be used between folder names.</p> <p>Please note the following:that</p> <ul style="list-style-type: none"> Backslashes must be escaped with another backslash in JSON. <p>If you are including the JSON into code when implementing a RESTful client, further escaping of the backslashes required by JSON may be required for code execution. You might require setting four backslashes between folder names.</p> <ul style="list-style-type: none"> It is not possible to upload files to the IDOC: root. The file name in the internal document selector does not have to be identical with the name of the file that is uploaded.
"Name": " FileName.FileExtention "	Mandatory	<p>The name of the file that shall be created in IDOC. The name does not have to be identical to the name of the file providing the content for upload. The file extension must match the uploaded content.</p> <p>Please note that the upload will fail if a file with the same name already exists in the defined Internal Document Selector folder and the field <code>OverwriteExistingFile</code> is missing or set to <code>false</code>.</p>
"OverwriteExistingFile": "true false"	Optional	<p>If this field is set to <code>true</code>, an existing file with the same name in the same folder of the Internal Document Selector will be overwritten by the uploaded file. If set to <code>false</code>, an existing file cannot be overwritten and upload will fail if a file with the same name already exists in the defined Internal Document Selector folder. The default value is <code>false</code>.</p>

Return value:

If the upload is successful, the return value will be a JSON object returning the IDOC ID if the uploaded document in a field `IDocID`.

Checking Whether the Alfabet Components Are Running

This endpoint can be used to check whether the Alfabet components that might be involved in the execution of a RESTful service request are running and can be accessed. The service call checks the availability of the following components:

- The Alfabet Web Application.
- The database server hosting the Alfabet database,
- If the server alias of the Alfabet Web Application is configured to connect to an Alfabet Server, the availability of that Alfabet Server is checked.

Endpoint name: `monitor`

HTTP method: `GET`

Service call:

`ServerAdress/api/monitor`

Header Fields:

`Authorization:TypeValue`



For information about the required Authorization header content, see [Authorization](#).

`Content-Type: application/json; charset=utf-8`

Payload

No payload required.

Return value:

The return value is a JSON object that informs about the availability of the components.



For example:

```
{
  "Name": "Alfabet WebApplication",
  "State": "Error",
  "Reason": "Alfabet WebApplication is running Database is running Alfabet Application Server error: Unable to connect to
the application server. Check the network connections and resources. Contact your administrator."
}
```

Chapter 7: Accessing the Alfabet User Interface from the External Application

The Alfabet RESTful API provides access to data in the Alfabet database to external applications processing the data. If the external application should also provide links to the Alfabet user interface to its users, links opening defined views of the Alfabet user interface can be defined using a special link syntax. The links point to either an object profile, object cockpit, or graphic view for a specified object. In the link, the `REFSTR` value of the object for that the view is opened must be provided. The `REFSTR` value of a relevant object can be requested via the Alfabet RESTful API in order to build the link. The endpoint `objects` can deliver a list of `REFSTR` values for all objects found via a query in a configured Alfabet report.

The required link syntax and the access rights that apply for links to the Alfabet user interface from external applications are described in detail in the section *Links to Alfabet Views from External Applications* in the reference manual *System Administration*.

Chapter 8: Testing the Alfabet RESTful API

For first tests of the Alfabet RESTful API, requests can be sent via a commercial REST client like Google's Advanced REST client or a Swagger editor.

For both the tests via a commercial REST client or tests of customer developed REST clients, tests for service calls for the endpoint update should only be performed on Alfabet test databases and not on the productive Alfabet database.

This chapter provides information about how to perform a simple test of a service call and how to interpret error codes that you might see during testing:

- [Testing the Alfabet RESTful API](#)
 - [Configurations Required to Use a Swagger Editor for Testing](#)

Testing the Alfabet RESTful API

You can use commercially available REST clients for simple tests of service calls to the Alfabet RESTful API. This does not require any additional configurations except for Swagger editors.

Configurations Required to Use a Swagger Editor for Testing

A Swagger editor can be used for testing. To view all calls in ready-to-use Swagger format, the Alfabet RESTful services and the Web browser must be configured as follows:

- 1) Open the Alfabet Administrator.
- 2) Click the **Alfabet Aliases** node in the explorer. A workspace with a toolbar opens.
- 3) In the toolbar, click **Tools > Configure alfabet.config**. An editor opens.
- 4) Click the Browse  button on the right of the **Web Folder** field and select the main directory of the Alfabet Web Application from the directory browser. The `alfabet.config` file in the subdirectory `config` of the selected directory opens in the editor.
- 5) Add the following code as child element of the `alfaSection` XML element, substituting `webApplicationFolderPath` with the absolute path to the virtual directory of the Alfabet Web Application:


```
<add key="SwaggerSpecFileName"
      value="WebApplicationFolderPath\SwaggerSpec\AlfabetWeb5_SwaggerSpec.json" />
```
- 6) Click **Save**. The change is saved and the editor is closed.
- 7) Close the Alfabet Administrator.
- 8) Go to the URL `http:// URLOfTheAlfabetWebApplication /swagger/docs/v2` and copy the JSON code that is displayed.
- 9) Open your Swagger editor and paste the JSON code into the editor code field.
- 10) Make sure that **Cross Origin Resource Sharing** is enabled on your browser.



The header: Access-Control-Allow-Origin: * must be included in the calls generated by the Sever. Enabling Cross Origin Resource Sharing in the browser is a simple way of achieving this if not otherwise implemented in the system.

Index

access permission	
overview	24
per endpoint	15
per mandate	21
per object.....	22
per object class	18
activation	9
ADIF export	
start.....	98
ADIF import	
start.....	95
ADIF scheme	
enabling for REST API	17
adifexport endpoint.....	98
adifimport endpoint.....	95
Alfabet components	
monitoring health.....	108
Alfabet metamodel	
reading culture settings	34
reading enumerations.....	43
reading object class model.....	34, 40
Alfabet object	
archiving	86
archiving and deleting	86
changing data.....	73
creating.....	73, 76
creating reference	84
creating relation.....	84
deleting	70
reading data	46
updating property values.....	80
Alfabet user	
anonymizing	92
regenerating password.....	89
Alfabet user interface	110
Alfabet.config	
JSON Web Token	10
max_api_requests_per_second	10
Allow Read via Rest API	18
Allow Write via Rest API	18
anonymizeuser endpoint.....	92
API Access Options	15
API Culture.....	28
archiveobject endpoint	
HTTP method.....	86
JSON body.....	86
return value	86
service call	86
authentication	
Web server setting	12
authorization	
API access options.....	15
client side	23
request	14
token request	23
user key.....	14
user password.....	14
availability check	
Alfabet Components.....	108
changing object data.....	80
class key	
reading from metamodel	34
reading metamodel structure	40
class setting	
REST access permission.....	18
classes endpoint	
HTTP method.....	40
parameter.....	40
return value	40
service call	40
configured report	
enabling for REST API.....	17
for endpoint objects.....	61
creating objects	
update endpoint	76
creating relations	
update endpoint	84
culture	

reading object data for culture.....	46	adifexport	98
culture settings		adifimport	95
reading from metamodel	34	anonymizeuser.....	92
database object		archiveobject.....	86
archiving	86	classes	40
archiving and deleting	86	delete	70
deleting	70	enums	43
reading data	46	idocdownload	104
date format	28	idocfilelist.....	101
delete endpoint		idocupload.....	105
HTTP method	70	metamodel	34
JSON body	70	monitor	108
return value	70	objects.....	46
service call.....	70	regeneratepassword	89
document		token	23
download	104	update	73
list	101	Versions	8
upload.....	105	workflow	94
download		enumeration	
internal document selector	104	reading metamodel structure	43
empty values		enums endpoint	
including in enums endpoint.....	43	HTTP method.....	43
including in metamodel endpoint.....	34, 40	parameter.....	43
emptyvalues parameter		return value	43
for endpoint classes	40	service call	43
for endpoint enums.....	43	file list	
for endpoint metamodel.....	34	internal document selector.....	101
Enable REST API v2.....	13	Generate API Password	15
endpoint		Google Advanced REST Client	111
		Has API V2 Access.....	15
		health monitoring	
		Alfabet components	108
		Http Header	
		authorization.....	14
		HTTP method	
		archiveobject endpoint	86
		classes endpoint	40
		delete endpoint.....	70
		enums endpoint.....	43
		metamodel endpoint.....	34
		objects endpoint.....	46
		regeneratepassword endpoint	89
		update endpoint	73
		IDOC	

download file	104	reading metamodel structure	34, 40
file list.....	101	object class property	
upload file	105	type ReferenceArray	84
idocdownload endpoint	104	object class	
idocfilelist endpoint.....	101	changing object data	73
idocupload endpoint	105	creating new object	76
Internal Document Selector		creating object.....	73
download file	104	updating property values.....	80
read content	101	object class property	
upload file	105	changing for object.....	73
JSON format		mandatory for new object.....	76
for creating relations	84	reading from metamodel	34
for updating object data.....	80	reading metamodel structure	40
in service call for update endpoint.	76, 80, 84	object data	
return value	31	exporting by report query	60
JSON request		exporting for objects by filter	52
for creating new objects	76	exporting for objects by REFSTR	47
JSON Web Token		exporting via query.....	61
changing.....	10	reading	46
key.....	14	reading different language versions.....	46
licenses	8	object in Alfabet Database	
limit		archiving.....	86
requests per second.....	10	archiving and deleting	86
link		changing data.....	73
to Alfabet user interface	110	creating	76
mandate	21	creating reference	84
max_api_requests_per_second.....	10	creating relation.....	84
Maximum number of requests per second		deleting.....	70
changing	10	reading data	46
metamodel endpoint		updating property values.....	80
HTTP method	34	object REFSTR	
parameter	34	exporting by report query	60
return value	34	objects endpoint	
service call.....	34	HTTP method.....	46
names parameter		objects by filter	52
for endpoint classes	40	objects by references	47
for endpoint enums.....	43	objects by report.....	60
object		parameter.....	46
archiving	86	return value	46
archiving and deleting	86	service call	46
creating.....	73	parameter	
deleting.....	70	emptyvalues	34, 40, 43
reading data for object.....	46	names	40, 43
object class		password.....	14
		regenerating for user.....	89
		permission.....	see access permission

Read permission	18	adifexport	98
ReadWrite permission.....	18	adifimport	95
Reference		containing object property values .	47, 52, 67
deleting	70	containing REFSTR list.....	69
ReferenceArray	84	containing report content	66
regeneratepassword endpoint		for endpoint enums	43
HTTP method	89	for endpoint metamodel	34, 40
JSON body	89	for endpoint objects.....	46
return value	89	for endpoint update	76, 84
service call.....	89	idocdownload	104
relation		idocfilelist.....	101
creating between objects.....	84	idocupload.....	105
deleting	70	monitor	108
RELATIONS table		objects by filter	52
deleting entry	70	objects by references	47
request		objects by report.....	60
limit	10	objects endpoint	46
requirements	see technical requirements	RESULT object	31
resetting		workflow	94
user password	89	Server Alias	
REST client		enable REST	13
for testing.....	111	server roles	11
implemenation for Alfabet REST API	6	service call	
RESTful service call			
limit for incoming	10		
return value			

adifexport	98	creating new objects	76
adifimport	95	creating relations	84
anonymizeuser endpoint	92	HTTP method	73
archiveobject endpoint	86	JSON body	73
authorization	23	return value	73
classes endpoint	40	service call	73
date format	28	updating object data	80
delete endpoint	70	updating objects	
enums endpoint	43	update endpoint	80
executing configured report	62	upload	
idocdownload	104	internal document selector	105
idocfilelist	101	user	
idocupload	105	anonymizing	92
metamodel endpoint	34	regenerating password	89
monitor	108	user based authorizationsee authorization	
objects by filter	52	user editor	
objects by references	47	REST settings	15
objects by report	60	user password	14
objects endpoint	46	user profile	
regenerate password endpoint	89	REST access permission	18
return value	31	view scheme	
syntax	31	REST access permission	18
time format	28	Web server	
update endpoint	73	authentication	12
workflow	94	Requirements	11
session cookies	6	server roles	11
Swagger	111	WebDAV	11
technical requirements		web.config	
licenses	8	handlers for RESTful services	9
required configuration	9	WebDAV	11
Web server	11	workflow	
time format	28	start	94
token	14, 23	workflow endpoint	94
update endpoint		workflow template	
		enabling for REST API	17
		Write permission	18