Web Services for Alfabet

Alfabet Reference Manual

Documentation Version Alfabet 10.15.1

Conventions used in the Documentation

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

Table of Contents

Chapter 1: Working with the Alfabet Web Services

Alfabet Web Services are designed to allow direct access to the data stored in the Alfabet database from external programs that run on different platforms. Platform interoperability is based on SOAP protocol and offers the possibility to make remote procedure calls via HTTP to the Alfabet database from programs that are written using different programming languages and that run on different operating systems.

It is recommended to use Alfabet Web Services for continuous data integration to or from a third-party application only. One time or infrequent batch data export or import to the Alfabet database should be handled via the Alfabet Data Integration Framework (ADIF).

Alfabet provides two Web Services

- ***AlfaQueryWebService*** to read data from the Alfabet database
- ***AlfaUpdateDataWebService*** to apply changes to the Alfabet database

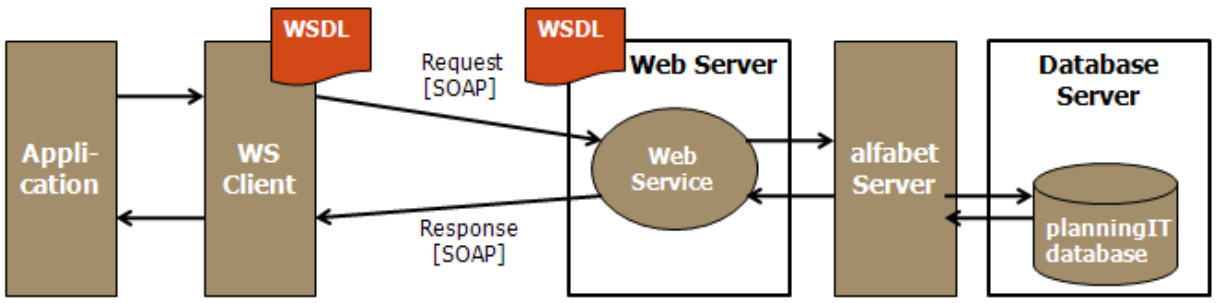
Using the Alfabet Web Services to Build an Interface for Reading From or Writing To the Alfabet database

The Alfabet Web Services must be implemented on a Web server to serve as an XML-based interface for access to the Alfabet database via external applications.

The contract between client and server is represented by Web Service Description Language (WSDL) which describes the methods that can be used to read or write data in the Alfabet database and the XML format that is returned by the methods.

Calls to the methods of the services from other programs or HTML forms via the Alfabet Web Services are based on SOAP format. On the server side, the Alfabet Web Services redirect calls to the running Alfabet application server and return XML documents as a result.

On the client side, the implementation of a Web service client is required for building an application on a Web service. The Web service client bridges the gap between the Web service procedures and the actual programming language of the external application. It assembles and disassembles the SOAP messages and services as proxy for the Web service methods.



You must do the following to use the Alfabet Web Services in order to access the Alfabet database:

- Set up the Alfabet Web Services on a Web server as a Web application.

  The setup procedure depends on the Web server used. Consult the documentation of your Web server for information about how to set up a Web service.

- Build a Web service client based on the WSDL of the Alfabet Web Services.

  The Web service client is usually built using a Web service client engine. For information, consult the documentation of the Web service client engine that will be implemented.

- Implement communication with the Web service client and the processing of data returned or sent via the Alfabet Web Services methods.

This document provides information about the Web service WSDL required to implement an external application based on the Alfabet Web Services and describes the methods used by the Alfabet Web Services and the XML output resulting from calls to the Alfabet Web Services.

When building a client application using the Alfabet Web services, take the following into account when implementing the external application:

- The calling program must handle exceptions that can occur in the service to control the success of the method call.
- The methods return plain XML in response to calls. The XML has to be transferred to in-memory data models within the client-side application.

  Special characters are escaped in the XML output. For more information about the available methods and the resulting XML output, see Methods Implemented for the Alfabet Web Services.

- The Alfabet Web Services do not check data integrity on the client and server side. It is the responsibility of the application architect to ensure data integrity.
- The Alfabet Web Services perform a simple authentication based on the database access mode either via Windows authentication or single-user authentication based on the database user name and password. Security can be enhanced by using the following methods:
  - Additional security mechanisms of the Web server (for example, encryption via HTTPS can be implemented).
  - The implementation of the client-side application can include additional security mechanisms like customer-access control via LDAP.

    For more information about the security mechanisms provided by the Alfabet Web Services, see Authentication of Alfabet Web Services.

Understanding the WSDL of the Alfabet Web Services

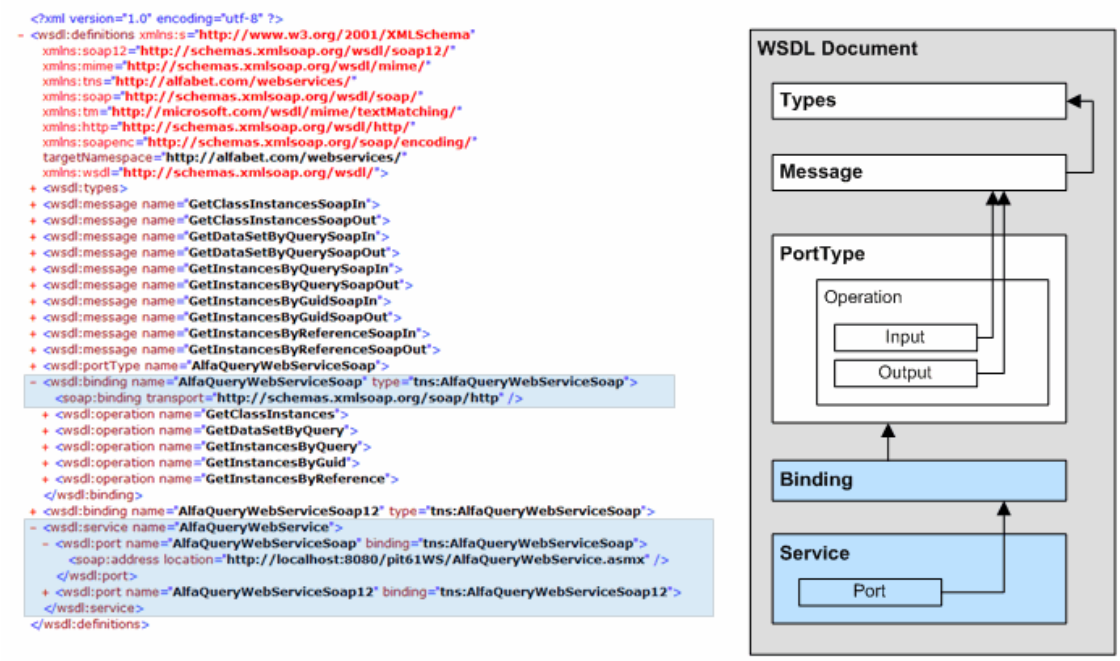
WSDL is the standard modeling language to describe Web services.

WSDL descriptions for the Alfabet Web Services are available in the standard interface of the Alfabet Web Services:

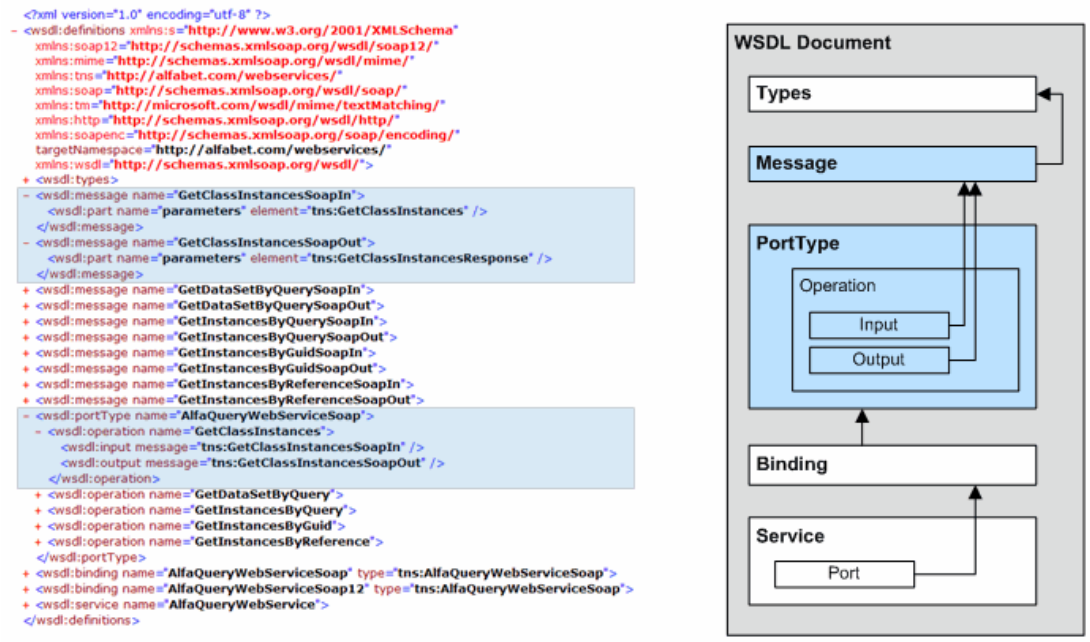
- ***AlfaQueryWebService.asmx?wsdl*** and
- ***AlfaUpdateDataWebService.asmx?wsdl***

The WSDL description is given in an XML document that is compliant with the W3C WSDL standard. The document contains information about:

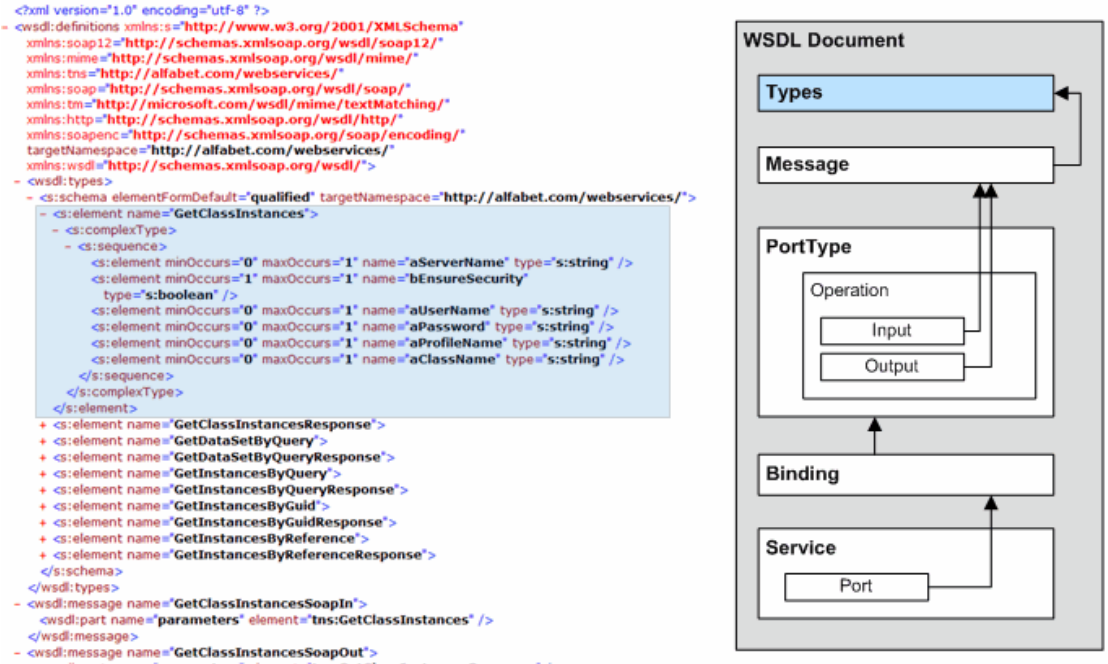
- the location of the service (`Service`, `Binding`) relevant for the SOAP based communication.



- supported operations and messages (`PortType`, `Messages`) relevant for the definition of the requests and responses provided via SOAP.



- method types (`Type`) describing the methods implemented with the Web services.



Most of the information in the XML is interpreted by the Web service client engine when building the Web service client. The information about the method calls are relevant for the implementation of the application on the client side and is described in detail in the section Methods Implemented for the Alfabet Web Services. The information includes the supported method calls, the arguments that must be submitted with the method calls, and the structure of the returned XML.

Chapter 2: Authentication of Alfabet Web Services

The Alfabet Web services connect to the Alfabet database via the Alfabet Server. Nevertheless, authentication is done directly on the database level providing a user name and password for login to the database on the database server level.

In case the authentication of the Web service client against the database is not desired or not available, a second mechanism is provided to ensure that existing clients can continue to operate. It is based on integrated Windows authentication between the application server and the database server. See Scenario II for a detailed description of the mechanisms involved in this authentication scenario.

The following sections provides a detailed overview on this processes and the involved mechanisms:

- Web Service Client Authentication
- Integrated Windows Authentication

Web Service Client Authentication

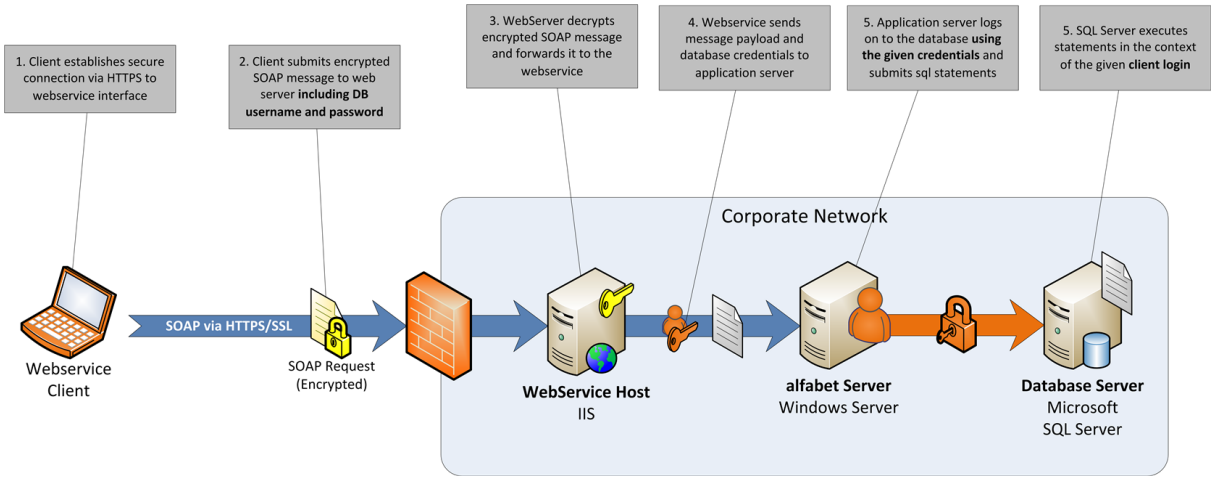
In this scenario, authentication against the database and hence the authorization of the request that has been submitted is encapsulated in the Web service client requests. Each Web service client type may operate using different credentials and hence may be provided with different permissions regarding available tables, views, or statements.

The SOAP protocol that is used to exchange messages between Web service clients and the Web server hosting the Alfabet Web Service does not include any means to encrypt sensitive information. Instead, this is the responsibility of the protocol being used on subordinate layers (For example, on the application layer or the transport layer). Software AG recommend that access to the Alfabet Web Service is provided exclusively via HTTPS to ensure secure communication.

The following list describes the processing of Web service requests using client authentication (see figure below):

1) The Web service client implementation composes a SOAP message containing all required parameters for the requested operation. This includes user name and password for the database account that is to be used to authenticate and authorize the execution of the required statements on the database.
2) Using the server certificate provided by the Web service host, the Web service client encrypts the SOAP message and transmits it to the Web service host in the body of the HTTP message.
3) The Web service host decrypts the body of the message and routes the SOAP message to the Web service request handler internal to Microsoft® Internet Information Services® for processing.
4) The Web service implementation unwraps the SOAP message and prepares a call to the corresponding application server interface providing the requested operation. It then submits the parameters of the requested operation as well as the credentials to the Alfabet Server.
5) The Alfabet Server uses the credentials that have been provided and tries to connect to the database server. If the connection attempt succeeds, the statements required to perform the requested operation are submitted to the database. The database then executes these statements using the provided identity and authorization. If no error has occurred, the data is sent back to the calling Web service host and ultimately returned to the requesting Web service client via HTTPS.

This scenario presumes that the Alfabet Server connects to the database server via client authentication and that Windows authentication is disabled at the database server. If the Alfabet Web Services use client authentication while the Alfabet Server uses Windows authentication for login to the database server, the Alfabet Server first connects to the database server using the user name and password provided with the Web service request to process a request from the Alfabet Web Services. But if login fails, the Windows Sign On of the Alfabet Server is used as a fall back and the connection is established even though no valid client authentication parameters have been provided by the Web service request.



Integrated Windows Authentication

In this scenario, the Web service client does not include any authentication information within the body of the request that is submitted to the Web server. Instead, authentication is performed by the application server providing the requested operation. That way, no sensitive database authentication information is disclosed to the Web service client side of the system. However, this approach does not enable a distinct definition of the permissions provided to different Web service client types. Instead, all Web service client calls will be executed in the context of the windows account used to operate the Alfabet Server service.

The following list describes the processing of Web service requests using Windows authentication:

1) The Web service client implementation composes a SOAP message containing all required parameter for the requested operation. The information contained in the body of the message exclusively consists of information required to perform the operation, e.g. request parameters. Authentication relevant parameters of the operation may be left blank.

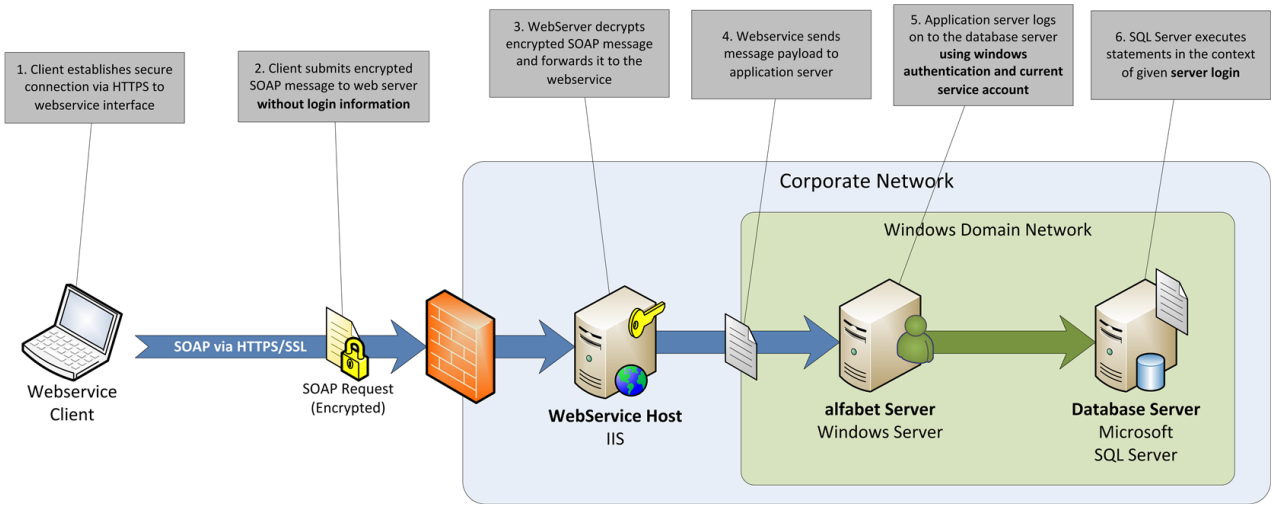
2) Using the server certificate provided by the Web service host, the Web service client encrypts the SOAP message and transmits it to the Web service host in the body of the HTTP message.

3) The Web service host decrypts the body of the message and routes the SOAP message to the Web service request handler internal to Microsoft® Internet Information Services® for processing.

4) The Web service implementation unwraps the SOAP message and prepares a call to the corresponding application server interface providing the requested operation. It then submits the parameters of the requested operation to the Alfabet Server.

5) Upon receiving the operation call, the Alfabet Server tries to connect to the database server using Windows integrated authentication. The account used for authentication against the database is the domain account under which the Alfabet Server service is running. If the connection attempt

succeeds, the statements required to perform the requested operation are submitted to the database. The database executes these statements using the Alfabet Server service account and with the associated permissions. In case no error has occurred, the data is sent back to the calling Web service host and ultimately returned to the requesting Web service client via HTTPS.



Chapter 3: Methods Implemented for the Alfabet Web Services

AlfaQueryWebService uses the following methods to read Alfabet data:

GetInstancesByQuery	returns objects with full data using an Alfabet query
GetInstancesByGuid	returns objects with full data using `GUID`s
GetClassInstances	returns all objects with full data of an object class defined by object class name
GetDataSetByQuery	returns objects (as references) using an Alfabet query
GetDataSetByNativeQuery	returns objects (as references) using a native SQL query

AlfaUpdateDataWebService uses the following method to manipulate Alfabet data:

UpdateData	inserts, updates and deletes objects and relations

All methods have 5 common arguments:

`aServerName`	The connection string (protocol, server, port and the name of remote server) used to connect the Web services to the relevant Alfabet Server: e.g. "tcp://localhost:8087/ Alfabet " syntax: tcp://(Machine):(Port)/(Server) (1) (Machine): the DNS name or IP address of the Alfabet Server host (2) (Port): the property "Port" defined in the configuration of the relevant remote alias (3) (Server): the attribute "Server" defined in the configuration of the relevant remote alias
`aUserName`	The user name for connection to the database server.
`aPassword`	The password for connection to the database server.
`aProfileName`	This argument is obsolete and available for backward compatibility reasons only. A string defined for this argument is ignored.
`bEnsureSecurity`	This argument has to match the **Ensure Security** setting defined in the server alias. Otherwise the calls to the Web service will fail. Set to `true` if **Ensure Security** is activated in the server alias configuration. Set to `false` if **Ensure Security** is deactivated in the server alias configuration.

Additional method-specific arguments are required for all available methods. These are specified in detail in the sections about the different methods. The detailed description of the general arguments is not repeated in the description of individual methods.

GetInstancesByQuery

```
GetInstancesByQuery (string aServerName, string aUserName, string aPassword,
string aQuery),
```

where

`aQuery` must be a valid Alfabet query.

For example, the Alfabet query could be:

```
"FIND Application WHERE Name like'A%'"
```

A detailed description of the Alfabet query language is provided in the chapter *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*. If you want to define queries in native SQL, or the Alfabet query language does not offer the query mechanisms that you want to implement, it is recommended that you use the method GetDataSetByNativeQuery instead or use the Web services offered by the underlying database platform instead of the Alfabet Web Services.

If more than one culture setting is defined for Alfabet, the query must include information about the culture setting that shall be used to return the results. For use in Web services only: The Alfabet query language offers a parameter to return results in a defined language. `LOCALE_ID` <language code> must be added to the Alfabet query prior to the `FIND` statement. For example, to return application names in the German language, you would define the following:

```
ALFABET_QUERY_500 LOCALE_ID 1031 FIND Application WHERE Name
like'T%' QUERY_XML &lt;QueryDef&gt; &lt;ShowProperty
Type='Property' ClassName='Application' Name='Name' /&gt;
&lt;/QueryDef&gt;
```

Return value: The method returns an XML document that contains instances found by an Alfabet query. The document provides information about all property values and relations of the instance stored in the Alfabet database.

The root element ***AlfaInstances*** has the following child elements:

- One element ***AlfaInstance*** for each instance found by the Alfabet query. The ***AlfaInstance*** element has one child element ***AlfaPropertyValue*** for each defined property of the instance.
- An element ***AlfaRelations*** with one child element ***AlfaRelation*** for each relation in the `RELATIONS` table of the Alfabet database that defines a relation from or to one of the instances found by the query.

The XML elements contain information about the object instances found by the Alfabet query in the following attributes:

Attribute	Description
AlfaInstance	
`Guid`	The `GUID` of the object class instance.
`ClassGuid`	The `GUID` of the object class.
`ClassName`	The Name of the object class.
`Mandates`	The mandate assignment of the object class. For more information about the mandate assignment for objects and the way the information is stored, see the reference manual *Alfabet Meta-Model*.
`Id`	The `Id` is extracted from the `REFSTR` property of the object class instance. The `REFSTR` is defined as: `<Id>-<ClassId>-<CompanyId>`
`ClassId`	The `ClassId` is extracted from the `REFSTR` property of the object class instance. The `REFSTR` is defined as: `<Id>-<ClassId>-<CompanyId>`
`CompanyId`	The `CompanyId` is extracted from the REFSTR property of the object class instance. The REFSTR is defined as: `<Id>-<ClassId>-<CompanyId>`
AlfaPropertyValue	
`Name`	The Name of the object class property.
`Type`	The data type of the object class property.
`Datum`	The value of the object class instance for the object class property.
AlfaRelation	

Attribute	Description
From	The GUID of the object class instance defining the relation with one of its properties.
To	The GUID of the object class instance to which the relation is established.
Property	The Name of the object class property establishing the relation.

```
<?xml version="1.0" encoding="utf-8" ?>
   <AlfaInstances>
      <AlfaInstance CompanyId="0"
         ClassName="ApplicationGroup"
         ClassGuid="F3790442ADFA41ACB153779CED6B24A3"
         Guid="10522BA133924E009D00B6C68AEF9001"
         ClassId="183"
         Id="5"
         Mandates="1">
            <AlfaPropertyValue Name="Name" Type="String"
            Datum="Group1" />
      </AlfaInstance>
      <AlfaRelations> <AlfaRelation
      From="F3790442ADFA41ACB153779CED6B24A3"
      To="841EBFD666654D21A02F6AF686416B15" Property="ConsistsOf"
      />
      </AlfaRelations>
</AlfaInstances>
```

GetInstancesByGuid

```
GetInstancesByGuid (string aServerName, string aUserName, string aPassword,
string aClassName, string guids)
```

where

aClassName must be a valid Alfabet class name.

guids comma-separated string of GUIDs.

Return value : The method returns an XML document that contains instances found by GUIDs. A description of the XML structure and the meaning of the attributes of the XML element in the document is given in the section GetInstancesByQuery.

```
<?xml version="1.0" encoding="utf-8" ?>
   <AlfaInstances>
      <AlfaInstance
         CompanyId="0"
         ClassName="ApplicationGroup"
         ClassGuid="F3790442ADFA41ACB153779CED6B24A3"
         Guid="10522BA133924E009D00B6C68AEF9001"
         ClassId="183"
         Id="5">
            <AlfaPropertyValue Name="Name" Type="String"
            Datum="Group1" />
      </AlfaInstance>
      <AlfaRelations>
         <AlfaRelation
            From="10522BA133924E009D00B6C68AEF9001"
            To="841EBFD666654D21A02F6AF686416B15"
            Property="ConsistsOf" />
      </AlfaRelations>
   </AlfaInstances>
```

GetClassInstances

```
GetClassInstances (string aServerName, string aUserName, string aPassword,
string aClassName)
```

where

aClassName must be a valid Alfabet class name

Return value : The method returns an XML document that contains all instances found in the object class with the Name defined in the argument ***aClassName***. A description of the XML structure and the meaning of the attributes of the XML element in the document is provided in the section GetInstancesByQuery.

```
<?xml version="1.0" encoding="utf-8" ?>
   <AlfaInstances>
      <AlfaInstanceCompanyId="0"
         ClassName="ApplicationGroup"
```



```
            ClassGuid="F3790442ADFA41ACB153779CED6B24A3"
            Guid="10522BA133924E009D00B6C68AEF9001"
            ClassId="183"
            Id="5">
                <AlfaPropertyValue Name="Name" Type="String"
                Datum="Group1" />
        </AlfaInstance>
        <AlfaRelations>
            <AlfaRelation
                From="10522BA133924E009D00B6C68AEF9001"
                To="841EBFD666654D21A02F6AF686416B15"
                Property="ConsistsOf" />
        </AlfaRelations>
    </AlfaInstances>
```

GetDataSetByQuery

```
GetDataSetByQuery (string aServerName, string aUserName, string aPassword,
string aQuery)
```

where

aQuery must be a valid Alfabet query

For example, an Alfabet query could be:

```
"FIND Application WHERE Name like'T%' QUERY_XML &lt;QueryDef&gt;
&lt;ShowProperty Type='Property' ClassName='Application' Name='Name' /&gt;
&lt;/QueryDef&gt;"
```

A detailed description of the Alfabet query language is provided in the chapter *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*. If you want to define queries in native SQL, you can use the method ***GetDataSetByNativeQuery*** instead. For more information, see GetDataSetByNativeQuery.

If more than one culture is defined for Alfabet, the query must include information about the culture that shall be used to return the results. For use in Web services only: The Alfabet query language offers a parameter to return results in a defined language. `LOCALE_ID` <language code> must be added to the Alfabet query prior to the `FIND` statement. For example, to return application names in the German language, you would define:

```
ALFABET_QUERY_500 LOCALE_ID 1031 FIND Application WHERE Name
like'T%' QUERY_XML &lt;QueryDef&gt; &lt;ShowProperty
Type='Property' ClassName='Application' Name='Name' /&gt;
&lt;/QueryDef&gt;
```


Return value: The method returns an XML document that contains the data set found by the Alfabet query:

Under the root element ***AlfaDataSet***, the following elements are included in the document:

- One ***AlfaDataSetCol*** element for each property included into the SHOW properties of the Alfabet query. The ***AlfaDataSetCol*** element has two child elements:
  - ***Name***: The name of the property that is displayed in the column.
  - ***Type***: The data type of the property that is displayed in the column.
- One ***AlfaDataSetRow*** element for each object class instance found by the Alfabet query. The ***AlfaDataSetRow*** element has multiple child elements:
  - **<column name>** : One element is added for each column defined with an ***AlfaDataSetCol*** element. The name of the element is identical to the name of the property. The content of the element is the value of the instance for the property.

```
<?xml version="1.0" encoding="utf-8" ?>
    <AlfaDataSet>
        <AlfaDataSetCol>
            <Name>Reference</Name>
            <Type>String</Type>
        </AlfaDataSetCol>
        <AlfaDataSetCol>
            <Name>Name</Name>
            <Type>String</Type>
        </AlfaDataSetCol>
        <AlfaDataSetRow>
            <Reference>183-5-0</Reference>
            <Name>Test1</Name>
        </AlfaDataSetRow>
        <AlfaDataSetRow>
            <Reference>183-6-0</Reference>
            <Name>Test2</Name>
        </AlfaDataSetRow>
        <AlfaDataSetRow>
            <Reference>183-12-0</Reference>
            <Name>Test4</Name>
        </AlfaDataSetRow>
    </AlfaDataSet>
```


GetDataSetByNativeQuery

```
GetDataSetByNativeQuery (string aServerName, string aUserName, string
aPassword, string aQuery)
```

where

aQuery must be a valid native SQL query

For example, an Alfabet query could be:

```
"SELECT app.REFSTR, app.NAME FROM APPLICATION app WHERE app.NAME like'A%'"
```

- The general rules for use of native SQL in the context of Alfabet are provided in the chapter *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*.
- The `SELECT` statement is evaluated in a different way than the general evaluation mode in Alfabet configurations. It is not required that the `REFSTR` is specified as the first argument in the `SELECT` statement. If defined, the `REFSTR` is written to the result data set.

This method expects a `SELECT` statement to be correctly executed. Please note however that it is not possible to check in the web service whether the query is defined with a SELECT statement instead of for example an `UPDATE` statement. To ensure that this is not leading to security issues, the Alfabet web services are protected by the authentication mechanisms described in the section Authentication of Alfabet Web Services.

Return value: The method returns an XML document that contains the data set found by the native SQL query.

Under the root element ***AlfaDataSet***, the following elements are included in the document:

- One ***AlfaDataSetCol*** element for each argument included in the `SELECT` statement of the native SQL query. The ***AlfaDataSetCol*** element has two child elements:
  - ***Name***: The name of the property that is displayed in the column.
  - ***Type***: The data type of the property that is displayed in the column.
- One ***AlfaDataSetRow*** element for each object class instance found by the Alfabet query. The ***AlfaDataSetRow*** element has multiple child elements:
  - **<column name>** : One element is added for each column defined with an ***AlfaDataSetCol*** element. The name of the element is identical to the name of the property. The content of the element is the value of the instance for the property.

```
<?xml version="1.0" encoding="utf-8" ?>
   <AlfaDataSet>
      <AlfaDataSetCol>
         <Name>Reference</Name>
         <Type>String</Type>
```



```
        </AlfaDataSetCol>
        <AlfaDataSetCol>
            <Name>Name</Name>
            <Type>String</Type>
        </AlfaDataSetCol>
        <AlfaDataSetRow>
            <Reference>183-5-0</Reference>
            <Name>Test1</Name>
        </AlfaDataSetRow>
        <AlfaDataSetRow>
            <Reference>183-6-0</Reference>
            <Name>Test2</Name>
        </AlfaDataSetRow>
        <AlfaDataSetRow>
            <Reference>183-12-0</Reference>
            <Name>Test4</Name>
        </AlfaDataSetRow>
    </AlfaDataSet>
```

UpdateData

```
UpdateData (string aServerName, string aUserName, string aPassword, string anXml)
```

where

anXml must be a valid XML document that contains update operations. Details are defined below.

The ***UpdateData*** method does not perform any checks on the Alfabet database. The Alfabet Server automatically sets the `REFSTR`, `ID` and `GUID` property of new objects and identifies objects via `GUID` and object classes via their Name attribute. When using the ***UpdateData*** method, you are responsible to ensure that no data integrity or accessibility problems occur and that mandatory properties are set for an object during creation. It is recommended that you consult the reference manual *Alfabet Meta-Model* in order to understand which settings are required for an object class.

The data in the XML document defined with the argument ***anXML*** must be XML compliant. No special characters are allowed in the XML content.

The XML document defined with the argument ***anXML*** must have a root element ***AlfaDataUpdate*** with one or multiple of the following child elements that are described in detail below:

Element	Action triggered by element
UpdateInstance	Changes the property values of existing objects in the Alfabet database.
CreateInstance	Creates a new object in the Alfabet database.
DeleteInstance	Deletes an object from the Alfabet database.
CreateRelation	Creates a relation between two existing objects in the Alfabet database.
DeleteRelation	Deletes a relation from the Alfabet database.

One ***AlfaDataUpdate*** element can contain multiple different child elements. If you define an XML document containing definitions for the manipulation of both instance data and relation data, the elements are processed in two steps:

- All child elements triggering the creation, update, and deletion of instances are processed first regardless of their position before or after elements for changing relation data.
- All child elements triggering the creation or deletion of relations are processed in a second step.

The two update steps are processed as separate update procedures. Therefore, if the change of relation fails, the processing of relations is rolled back while the instances created in the first step during instance processing are kept in the Alfabet database.

The following example shows a typical update XML document:

```
<AlfaDataUpdate>
    <UpdateInstance
        ClassName="Application"
        Guid="841EBFD298654D21A02F6AF686416B15">
            <AlfaPropertyValue Name="Name" Type="String"
            Datum="Test1"/>
            <AlfaPropertyValue Name="Status" Type="String"
            Datum="Active"/>
    </UpdateInstance>
    <CreateInstance
        ClassName="ApplicationGroup">
            <AlfaPropertyValue Name="Name"Type="String"
            Datum="Test4"/>
    </CreateInstance>
    <DeleteInstance
```



```
            ClassName="ApplicationGroup"
            Guid="841EBFD298654D21A02F6AF686416B15"/>
        <CreateRelation
            From="C6F3EEBE4595414DAF359EF6FA79C192"
            To="BBD25FA714D84ACD896A95FCCFCA1410"
            Property="ConsistsOf" />
        <DeleteRelation
            From="C6F3EEBE4595414DAF359EF6FA79C192"
            To="BBD25FA714D84ACD896A95FCCFCA1410"
            Property="ConsistsOf" />
    </AlfaDataUpdate>
```

Return value: The method returns an XML document that informs about the changes performed in the Alfabet database:

```
<?xml version="1.0"?>
    <AlfaDataUpdateResult
        InstancesUpdated="1"
        InstancesAdded="20"
        InstancesDeleted="3"
        RelationsAdded="3"
        RelationsDeleted="1340"/>
```

UpdateInstance

The ***UpdateInstance*** element defines one or multiple objects for that property values shall be updated in the Alfabet database. The Name attribute of the object class is used to identify the object class. A single object is identified via its `GUID` while multiple objects are identified via an Alfabet query.

Within the ***UpdateInstance*** element, the values for the properties that shall be updated are defined in child elements ***AlfaPropertyValue***.

Please note the following for the update of instance data:

- The `GUID`, `REFSTR`, and `ID` values for objects in the Alfabet database are automatically assigned to the object and must not be changed via an ***AlfaPropertyValue*** element.
- Dates must be specified in the format YYYY-MM-DD.
- It is recommended that you consult the reference manual *Alfabet Meta-Model* in order to understand the requirements of property settings and the restrictions that may apply to the setting of the correct value. For example, a property may be based on an enumeration and thus only values defined in the enumeration are applicable, or a class key may exist that requires that the value is unique within the database table.

- If the ***UpdateInstance*** element finds multiple instances via an Alfabet query, all instances are updated with the same property values defined in the child elements ***AlfaPropertyValue***. Ensure that the properties definitions in the ***UpdateInstance*** element do not contain properties that are part of a unique key configuration.
- Properties of the type `ReferenceArray` with the attribute **ReferenceSupport** set to `true` define relations that are stored in the `RELATIONS` table of the Alfabet database. These properties cannot be defined via the ***AlfaPropertyValue*** element of the ***UpdateInstance*** element. Instead, ***CreateRelation*** and ***DeleteRelation*** elements must be defined to update the relations. For more information, see the sections CreateRelation and DeleteRelation.

UpdateInstance Element Identifying Objects via GUID

The following attributes must be defined for the element ***UpdateInstance*** and the child elements ***AlfaPropertyValue***:

Attribute	Description
UpdateInstance	
`ClassName`	The value of the **Name** attribute of the object class for that the instance shall be created.
`Guid`	The `GUID` of the object class instance that shall be updated
AlfaPropertyValue	
`Name`	The value of the **Name** attribute of the object class property for that a value is defined with this ***AlfaPropertyValue*** element.
`Type`	The value of the **Type** attribute of the object class property for that a value is defined with this ***AlfaPropertyValue*** element.
`Datum`	The value to be set for the property.

```
<UpdateInstance
    ClassName="Application"
    Guid="841EBFD298654D21A02F6AF686416B15">
```



```
        <AlfaPropertyValue Name="Name" Type="String" Datum="Test1"/>
        <AlfaPropertyValue Name="Status" Type="String"
        Datum="Active"/>
</UpdateInstance>
```

The Alfabet application server will attempt to find an existing instance with a given `GUID`. If the instance exists, the corresponding values are updated and saved. Otherwise, a new instance will be created using the given `GUID`, and the values will be set and saved.

UpdateInstance Element Identifying Objects Via an Alfabet Query

The following attributes must be defined for the element ***UpdateInstance*** and the child elements ***AlfaPropertyValue***:

Attribute	Description
UpdateInstance	
`Query`	An Alfabet query that returns the objects to be updated.
AlfaPropertyValue	
`Name`	The value of the Name attribute of the object class property for that a value is defined with this ***AlfaPropertyValue*** element.
`Type`	The value of the Type attribute of the object class property for that a value is defined with this ***AlfaPropertyValue*** element.
`Datum`	The value to be set for the property.

```
<UpdateInstance
    Query="FIND Application
    WHERE (AND Name like'T%' Status ='Retired'">
        <AlfaPropertyValue Name="Status" Type="String"
        Datum="Active"/>
</UpdateInstance>
```

For all instances found by the Alfabet query, the corresponding values will be updated and saved.

Note the following for the definition of the Alfabet query:

- The definition of `SHOW` properties is not required for the Alfabet query.
- Objects of the class defined in the `FIND` statement are updated. `JOIN`s can be defined to specify `WHERE` conditions based on references to other object classes, but objects of classes added to the results via a `JOIN` are not updated.
- A detailed description of the Alfabet query language is provided in the chapter *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*. If you want to define queries in native SQL or the Alfabet query language does not offer the query mechanisms that you want to implement, it is recommended that you use the Web services offered by the underlying database platform instead of the Alfabet Web Services.

CreateInstance

The ***CreateInstance*** element defines the object class for which a new element shall be added to in the Alfabet database. The **Name** attribute of the object class is used to identify the object class.

Within the ***CreateInstance*** element, the values for the properties of the new object are defined in the child element ***AlfaPropertyValue***.

The following attributes must be defined for the elements ***CreateInstance*** and ***AlfaPropertyValue***:

Attribute	Description
CreateInstance	
`ClassName`	The value of the **Name** attribute of the object class for that the instance shall be created.
AlfaPropertyValue	
`Name`	The value of the **Name** attribute of the object class property for that a value is defined with this ***AlfaPropertyValue*** element.
`Type`	The value of the **Type** attribute of the object class property for that a value is defined with this ***AlfaPropertyValue*** element.
`Datum`	The value to be set for the property.

```
<CreateInstance
   ClassName="Application">
      <AlfaPropertyValue Name="Name" Type="String" Datum="Test1"/>
      <AlfaPropertyValue Name="Version" Type="String" Datum="2.1"/>
      <AlfaPropertyValue Name="StartDate" Type="Date" Datum="2012-
      01-28"/>
      <AlfaPropertyValue Name="EndDate" Type="Date" Datum="2014-01-
      28"/>
</CreateInstance>
```

Please note the following for the creation of new instances:

- The `GUID`, `REFSTR`, and `ID` values for the new objects are automatically assigned to the object and must not be specified in an ***AlfaPropertyValue*** element.
- Dates must be specified in the format YYYY-MM-DD.
- Some of the properties for an object class are mandatory. These properties must be set with an ***AlfaPropertyValue*** element.
- It is recommended that you consult the reference manual *Alfabet Meta-Model* in order to understand the requirements of property settings and the restrictions that may apply to the setting of the correct value. For example, a property may be based on an enumeration and thus only values defined in the enumeration will be applicable, or a class key may exist that requires that the value is unique within the database table.
- Properties of the type `ReferenceArray` for which the **ReferenceSupport** attribute is set to `true` define relations that are stored in the `RELATIONS` table of the Alfabet database. These properties cannot be defined via ***AlfaPropertyValue*** element of the ***CreateInstance*** element. Instead, ***CreateRelation*** elements must be defined to create the relations. For more information, see CreateRelation.

DeleteInstance

The ***DeleteInstance*** element defines one or multiple objects that shall be deleted from the Alfabet database. The **Name** attribute of the object class is used to identify the object class. A single object is identified via its `GUID` while multiple objects are identified via an Alfabet query.

Please note the following for the deletion of instances:

- When an object is deleted, the Alfabet Server automatically applies cascade deletion of the instances that are related to each other via properties defined in the **Integrity Info** attribute of the object class of the deleted object.
- If the instance is related to other objects in the Alfabet database s via a relation that is stored in the `RELATIONS` table of the Alfabet database, and is not part of the **Integrity Info** attribute, the relations must also be deleted. Relations are deleted via ***DeleteRelation*** elements. For more information, see DeleteRelation.

DeleteInstance Element Identifying Objects via GUID

The following attributes must be defined for the element ***DeleteInstance***:

Attribute	Description
`ClassName`	The value of the **Name** attribute of the object class for that the instance shall be created.
`Guid`	The `GUID` of the object class instance that shall be deleted.

```
<DeleteInstance
    ClassName="Application"
    Guid="841EBFD298654D21A02F6AF686416B15"/>
```

DeleteInstance Element Identifying Objects Via an Alfabet Query

The following attributes must be defined for the element ***DeleteInstance***:

Attribute	Description
`Query`	An Alfabet query that returns the objects to be deleted.

```
<DeleteInstance
    Query="FIND Application WHERE (AND Name like'T%' Status
    ='Retired'"/>
```

Note the following for the definition of the Alfabet query:

- The definition of `SHOW` properties is not required for the Alfabet query.
- Objects of the class defined in the `FIND` statement or deleted. `JOIN`s can be defined to specify `WHERE` conditions based on references to other object classes but objects of classes added to the results via a `JOIN` are not deleted.
- A detailed description of the Alfabet query language is provided in the chapter *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*. If you want to define queries in native SQL or the Alfabet query language does not offer the query mechanisms that you want to implement, it is recommended that you use the Web services offered by the underlying database platform instead of the Alfabet Web Services.

CreateRelation

The ***CreateRelation*** element triggers the creation of one or multiple relations in the Alfabet database.

The ***CreateRelation*** element creates relations stored in the `RELATIONS` table of the Alfabet database. These are relations based on a property of the type `ReferenceArray`.

The ***CreateRelation*** element can either identify objects on the basis of their `GUID` property or by means of an Alfabet query. The identification of objects via `GUID`s requires a separate ***CreateRelation*** element for each relation to be created. When objects are identified via an Alfabet query, one ***CreateRelation*** element can trigger the creation of multiple relations.

CreateRelation Element Identifying Objects Via Guid

The ***CreateRelation*** element must have the following attributes:

Attribute	Description
`From`	`GUID` of the object class instance in the Alfabet database that establishes the relation via the property defined in the attribute `Property`.
`To`	`GUID` of the object class instance in the Alfabet database to that the relation is established.
`Property`	**Name** attribute of the object class property establishing the relation.

The Alfabet Server searches for instances with the specified `GUID`s and creates the relation between them if the instances have been found.

```
<CreateRelation
   From="C6F3EEBE4595414DAF359EF6FA79C192"
   To="BBD25FA714D84ACD896A95FCCFCA1410"
   Property="ConsistsOf" />
```

CreateRelation Element Identifying Objects Via alfabet Query

The ***CreateRelation*** element must have the following attributes:

Attribute	Description
QueryFrom	An Alfabet query returning object class instances in the Alfabet database that establish the relation via the property defined in the attribute `Property`.
QueryTo	An Alfabet query returning object class instances in the Alfabet database to that the relations are established.
Property	**Name** attribute of the object class property establishing the relation.

```
<CreateRelation
   QueryFrom="FIND Project WHERE Name like'CMS_%'"
   QueryTo="FIND Project WHERE Name ='CMSImplementation'"
   Property="Parent" />
```

The Alfabet Server searches for instances via the defined queries and creates the relation between them if the instances have been found.

Note the following for the definition of the Alfabet query:

- The definition of `SHOW` properties is not required for the Alfabet query.
- Objects of the class defined in the `FIND` statement are considered. `JOIN`s can be defined to specify `WHERE` conditions based on references to other object classes but objects of classes added to the results via a `JOIN` are not considered.
- A detailed description of the Alfabet query language is provided in the chapter *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*. If you want to define queries in native SQL or the Alfabet query language does not offer the query mechanisms that you want to implement, it is recommended that you use the Web services offered by the underlying database platform instead of the Alfabet Web Services.

DeleteRelation

The ***DeleteRelation*** element triggers the deletion of one or multiple relations from the Alfabet database.

The ***DeleteRelation*** element deletes relations stored in the `RELATIONS` table of the Alfabet database. These are relations based on a property of the type `ReferenceArray`.

The ***DeleteRelation*** element can either identify objects on basis of their `GUID` property or by means of an Alfabet query. The identification of objects via `GUID`s requires a separate ***DeleteRelation*** element for each relation to be deleted. When objects are identified via an Alfabet query, one ***DeleteRelation*** element can trigger the deletion of multiple relations.

DeleteRelation Element Identifying Objects Via Guid

The ***DeleteRelation*** element must have the following attributes:

Attribute	Description
`From`	`GUID` of the object class instance in the Alfabet database that establishes the relation via the property defined in the attribute `Property`.
`To`	`GUID` of the object class instance in the Alfabet database to that the relation is established.
`Property`	**Name** attribute of the object class property establishing the relation.

The Alfabet Server searches for instances with the specified `GUID`s and removes the relation between them if the instances have been found.

```
<DeleteRelation
    From="C6F3EEBE4595414DAF359EF6FA79C192"
    To="BBD25FA714D84ACD896A95FCCFCA1410"
    Property="ConsistsOf" />
```

DeleteRelation Element Identifying Objects Via an Alfabet Query

The ***DeleteRelation*** element must have the following attributes:

Attribute	Description
`QueryFrom`	An Alfabet query returning object class instances in the Alfabet database that establish the relation via the property defined in the attribute `Property`.
`QueryTo`	An Alfabet query returning object class instances in the Alfabet database to that the relations are established.
`Property`	**Name** attribute of the object class property establishing the relation.

```
<DeleteRelation
   QueryFrom="FIND Project WHERE Status ='New'"
   QueryTo="FIND Project WHERE Status ='Discarded'"
   Property="Parent" />
```

The Alfabet Server searches for instances via the defined queries and removes the relation between them if the instances have been found.

Note the following for the definition of the Alfabet query:

- The definition of `SHOW` properties is not required for the Alfabet query.
- Objects of the class defined in the `FIND` statement are considered. `JOIN`s can be defined to specify `WHERE` conditions based on references to other object classes but objects of classes added to the results via a `JOIN` are not considered.
- A detailed description of the Alfabet query language is provided in the chapter *Defining Queries* in the reference manual *Configuring Alfabet with Alfabet Expand*. If you want to define queries in native SQL or the Alfabet query language does not offer the query mechanisms that you want to implement, it is recommended that you use the Web services offered by the underlying database platform instead of the Alfabet Web Services.

Index

