A B A C U S

ABACONNECT WEBSERVICES

TECHNICAL OVERVIEW

Using AbaConnect WebServices

Version 2022

January 2022/KS

Table of Contents

Introduction

This documentation is intended for developers, or partners, who are developing applications with AbaConnect WebServices. The documentation can also be useful for non-developers who are interested in obtaining a better understanding of the AbaConnect WebServices functionality.

AbaConnect is a general term describing the ABACUS Application interfaces, used for transferring data to and from ABACUS Software applications. The interfaces are categorized under the Application. Data can be exported or imported via most AbaConnect interfaces in the following ways :

- via the ABACUS User Interface using Prog 625 AbaConnect – data can be exported and imported via XML and/or ASCII file formats.

- via Commandline using the available ABACUS programs – data can be exported and imported via XML and/or ASCII file formats.

- via WebService – most of the major ABACUS interfaces also supports the transfer of data via WebServices.

AbaConnect WebServices are an extension of the ABACUS AbaConnect Framework. Data can be transferred via AbaConnect WebServices for selected ABACUS AbaConnect interfaces.

Generally, all main AbaConnect interfaces are available as WebServices. ABACUS Application teams can configure their interfaces within the AbaConnect Framework. The Application teams can decide if they make and interface available via WebService, or not. To check if an AbaConnect interface is available as WebService, please refer to the ABACUS download page on the ABACUS Homepage :
http://www.abacus.ch/de/downloads-page/abaconnect

If the WSDL is available for an AbaConnect interface then is can be downloaded from the ABACUS Homepage. The ABACUS AbaConnect WSDL's are not available via the URL of the WebService Endpoint itself. The WSDL's can only be downloaded via the ABACUS Homepage.

AbaConnect Documentation

AbaConnect documentation is available on the ABACUS Homepage (http://www.abacus.ch). There are 2 main AbaConnect areas available :

- AbaConnect Interface Documentation :
  http://www.abacus.ch/de/downloads-page/abaconnect/dokumentationen
  contains the AbaConnect interface documentation, FAQ's and Commandline documentation

- AbaConnect WebServices
  http://www.abacus.ch/de/downloads-page/abaconnect/webservices
  contains the AbaConnect WSDL's, WebService documentation and examples in

C#.NET, VB.NET and Java

N.B. The AbaConnect interface documentation is also valid for the WebService. The data structure used for file-based XML import and export is identical to that used by AbaConnect WebServices. The AbaConnect Mapping used for WebServices is fixed to the AbaDefault Mapping structure. It is not possible to reference user-defined Mappings with the AbaConnect WebServices.

WebService SOAP Overview (*Simple Object Access Protocol*)

SOAP is a standard protocol used by WebServices to transfer information between client and server systems. The SOAP standrad is defined by World Wide Web Consortium, W3C (http://www.w3.org/TR/soap). The main concept of SOAP is that data is transferred in packets called Envelopes. The data is generally transferred using HTTP transport protocols, much the same as used when internet browsers transfer data for Homepages. The SOAP messages are packets of XML-formatted data. Typically, the communication consists of a request and a response. For every request there is a corresponding response.

Example of typical AbaConnect SOAP Message :

Request :

```
POST /abaconnect/services/Address_2010_00 HTTP/1.0
SOAPAction: "login"
User-Agent: Axis2
Host: localhost:8888
Content-Length: 634
Content-Type: text/xml; charset=UTF-8

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header></SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns1:Login xmlns:ns1="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
      <ns1:UserLogin>
        <ns1:UserName>Administrator</ns1:UserName>
        <ns1:Password>eli</ns1:Password>
        <ns1:Mandant>7777</ns1:Mandant>
      </ns1:UserLogin>
    </ns1:Login>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response :

```
HTTP/1.1 200 OK
Server: AbaSioux/2010.00
Connection: keep-alive
Date: Tue, 18 Dec 2012 13:47:50 GMT
Last-Modified: Tue, 18 Dec 2012 13:47:50 GMT
Expires: Tue, 18 Dec 2012 13:47:50 GMT

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <LoginResponse xmlns="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes"
xmlns:act="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
      <act:LoginToken>53ebec0edb73498c893b2ffd77ff8ac88af14e50</act:LoginToken>
      <act:Code>0</act:Code>
      <act:Message></act:Message>
    </LoginResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

A SOAP structure must always contain a root element "Envelope" and contain an element

"Body". The contents of the "Body" element are defined by the implementation. The AbaConnect specific information is contained within the "Body" element.

The SOAP Messages are generally generated automatically by the software components (e.g. .NET and JAVA). When using the AbaConnect WSDL's and development tools such as .NET and JAVA, the SOAP Messages will be automatically generated. The developer can then concentrate on using the functionality available in the interface. The AbaConnect interfaces have a maximum of 12 operations available.

Functionality of AbaConnect WebServices

The basic functionality of an AbaConnect WebService interface is defined by operations. Operations define the methods that are available for the AbaConnect WebService.

All AbaConnect interfaces contain the following 6 operations :

- **Ping** - test if interface is available (does not require Login information)
- **Login** – login to ABACUS via AbaConnect WebService interface
- **Logout** – logout from ABACUS via AbaConnect WebService interface
- **IsFinished** – request to see if a data operation has been completed
- **InteroperabilityTest** – test-operation to test fomat compatibility
- **IsAlive** – checks to see if a user session is active

The following operations are optional and may also be implemented in an AbaConnect WebService interface :

Import Operations :

- **Insert** – inserts a new data record
- **Update** – updates an existing data record
- **Save** – saves a data record, combination of Insert and Update
- **Delete** – deletes an existing data record

Export Operations :

- **Find** – searches for existing data records
- **DefaultValues** – retrieves a default data structure

Licence Options for AbaConnect

AbaConnect is a framework that is installed and available with the Abacus installation. Some of the interfaces require a licence option to export or import data via the interface. These options are part of the Abacus application configuration. More information about the available licence options can be found in the Abacus price lists. There is no separate licence option required to use the AbaConnect framework.
If an AbaConnect interface requires a licence option, this licence option is required for export and import of data via the AbaConnect WebServices (e.g. for Find-Requests and Save-Request). Note in the Abacus UI (Prog 625) is may be possible to export selected master data interfaces (such as address, customer and supplier) in an XML file, without the interface licence option.

Ping Operation

The Ping operation allows the WebService Client to test the connection to the ABACUS Server without the need to login (or before the login). The Ping request contains a string

type parameter. A simple string ("Hello AbaConnect"), can be sent to the Server and will be returned in a reponse string.

Example of a SOAP PingRequest :

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header></SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <ns1:PingRequest xmlns:ns1="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
         <ns1:Echo>Hello AbaConnect</ns1:Echo>
      </ns1:PingRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example of a SOAP PingResponse

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header></soapenv:Header>
   <soapenv:Body>
      <act:PingResponse xmlns:act="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
         <act:Echo>hello this is the AbaConnect Webservice Server! Your message was: Hello AbaConnect ( Ping-Nr: 1)</act:Echo>
      </act:PingResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

There are 2 additional features for a PingRequest that may be useful for some AbaConnect Webservice applications.

1. **Ping Exception with empty string** – if an empty text string is sent via the Ping request, the ABACUS Server will return an Exception. This allows developers to test the functionality of Exception handling in the WebService Client application.

Example of a SOAP PingRequest with empty Echo text :

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header></SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <ns1:PingRequest xmlns:ns1="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
         <ns1:Echo></ns1:Echo>
      </ns1:PingRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example of an AbaConnectFaultException generated from an empty Ping Echo text

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>soapenv:Server</faultcode>
      <faultstring>AbaConnectFaultException</faultstring>
      <detail>
        <act:AbaConnectFault xmlns:act="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
          <act:Code>4711</act:Code>
          <act:Message>This is a Test Exception from the AbaConnect Webservice</act:Message>
          <act:Cause>message from the cause</act:Cause>
        </act:AbaConnectFault>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

2. **Ping to retrieve ABACUS Version** – if the following text "[[version]]" is contained in the Ping Request string, the ABACUS Server will return the installed ABACUS Version and CD-DATE. The format of the ABACUS Version is V[[yyyy.xx]] and the CDDATE[[yyyy-MM-dd]]. There are examples of these requests shown in the AbaConnect WebService examples.

Example of a SOAP PingRequest containing the [[version]] keyword in the Echo text :

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header></SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <ns1:PingRequest xmlns:ns1="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
         <ns1:Echo>Hello AbaConnect [[version]]</ns1:Echo>
      </ns1:PingRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example of a SOAP PingResponse containing the [[version]] keyword ABACUS Version and CD-DATE.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header></soapenv:Header>
   <soapenv:Body>
      <act:PingResponse xmlns:act="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
         <act:Echo>hello this is the AbaConnect Webservice Server! Your message was: Hello AbaConnect [[version]]( Ping-Nr: 1)
V[[2015.201]] CDDATE[[2015-12-31]]</act:Echo>
      </act:PingResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Login Operation

A login must be made before any other operation involving data transactions can be made via the AbaConnect interfaces. The normal login for WebServices uses a normal ABACUS User and consists of :

1. Username – A valid user name defined in the ABACUS system.
2. Password – A valid password for the specified user.
3. Mandant – The client/mandant number defined in ABACUS.

After a successful Login, the user session will remain active until a logout operation (* refer to note below on User Timeouts). The Login operation requires the username, password and Mandant to identify the user. A successful login will return a valid login token that can be used for further operations to identify the current user session.

The active Mandant for the WebService Session is defined during the login with the Mandant number. To change the active Mandant it is necessary to logout from the curretn Session and login with a different Mandant.

Hint : The AbaConnect WebService Source Code examples in JAVA and .NET provide good examples of the use of Login and Logout.
The AbaConnect WebService examples are available on the ABACUS Hompage :
http://www.abacus.ch/de/downloads-page/abaconnect/webservices

Note : It is important that the login and logout sequences are cleanly handled by the client applications. Incorrect handling of the user login and logout process may barre the user for further operations. There should be a Login before starting any data requests via the AbaConnect interfaces and a Logout when the requests are finished. Requests can consist of multiple calls using one or more AbaConnect interfaces. It is not necessary to login separately for each AbaConnect interface. For example, the Login Token returned from the login in one interface can be used to identify the current user over one or more AbaConnect interfaces.

If an AbaConnect WebService Login is called with an ABACUS User that is already logged in via AbaConnect WebService, the actual User Session will be returned. Multiple User Sessions for the same User are not possible.

It is not possible to Login via the ABACUS Menu and AbaConnect WebServices simultaneously, with the same User. If a User is logged in via the ABACUS Menu, the Login with the same User via WebService will be block until the User logs out via the ABACUS Menu. If a User logs in via WebService, and later, attempts to log in with the same User via the ABACUS Menu, a message box will be displayed in the ABACUS Menu. The User login via the ABACUS Menu has the option to override the current WebService Login Session and create a new Login Session via the ABACUS Menu. In this case the User Login via the ABACUS Menu will create a new Login Session and the existing WebService User Session will be terminated, and further requests via the WebService will generate errors that the User Session is no longer valid.

As of Abacus V2018 the user names used for the login via AbaConnect WebService must be configured in a server configuration file. Only usernames configured in the server configuration file can be used to login via the AbaConnect WebService. With this configuration possibility specific users can be specified, that are permitted to access the AbaConnect WebServices.
The AbaConnect WebService Users are listed in the "abasystem.properties" file with the keyword "AbaConnect.WebService.Users" containing a list of the user names.

For Example :
In the file : x:\abac\system\abasystem.properties
AbaConnect.WebService.Users=”Administrator”,”abaconnect”

would allow the Users "Administrator" and "abaconnect" access to the AbaConnect WebServices. The user names are separated with a comma “,” or colon “:”. In the rare case that a user name contains a comma “,”, the colon should be used to separate the user names.

Note : It is not recommended to use the "Administrator" for WebService access on a productive installation. For a normal Abacus user, the user will be disabled in the Abacus User Management when an incorrect password is used multiple times by the login attempt. A disabled user can be reactivated again by the "Administrator" in the Abacus User Management. The Administrator cannot be disabled.

Logout Operation

The Logout operation can be performed using the login token in the LoginType parameter to identify the current user. A logout should be made after all the required requests are finished. It is not recommended to leave user sessions open indefinitely, because the ABACUS User Session can be close after a period of inactivity, by the ABACUS System. A period of activity is normally defined as 30 to 60 minutes of inactivity. Logging out will ensure that the user session and any cached server memory is cleared.

If the login token is no longer known, it is also possible to logout using the username, password and Mandant to identify the user. It is recommended to use the login token

where possible.

User Session Management and Timeouts

The Login Session Management must be controlled from the WebService Client. The default behaviour for a Login via AbaConnect WebServices is that the User Session will be created on the ABACUS and remain active until the WebService Client logs out.

The user sessions are handled by the ABACUS server. If a user session is inactive for a longer period of time (e.g. 30 – 60 minutes), the ABACUS Server may close the user session automatically. If an AbaConnect WebService client program experiences long or indefinite periods of inactivity, then it is better to log out and login when the services are required again. Therefore, the WebService Client should logout after longer periods of inactivity to avoid having an inactive User Session closed (cleaned up) by the ABACUS Server.

A User Session will only be created once for the same user. The default behaviour for AbaConnect WebService is that the Login with the same Username and Password will adopt the current User Session, if one is already open. Therfore, multiple client instances or processes, should not use the same User. The WebService Requests should be handled by a single process, the the data request musts be handled sequentially. It is not possible to execute multiple data requests simultaneously, with a single User Session, via the AbaConnect WebServices.

As of ABACUS V2014 SP 20.09.2014 the default Login behaviour to adopt existing WebService Login Session can be controlled. The default behaviour, adopts the current User Session, when the User is already logged in via the WebService. With an additional Login Parameter the adoption of the current User Session can be rejected. This is useful when the WebService runs from multiple locations and may already be logged in from one of the locations with the same User. To use this feature an additional Login Parameter must be sent via the WebService.

Special Note : The additional Login Parameter can only be sent when using a newly generated WSDL file. The older WSDL files do not allow additional Parameters to be sent with a Login operation. All WSDL generated as of V2014, should automatically have the Login Parameters included in the WSDL files. Earlier WSDL files can be generated on request. This feature is only available as of ABACUS V2014 SP 20.09.2014. It can be used with all AbaConnect interface versions as of ABACUS V2014 SP 20.09.2014, provided that the WSDL for the particular interface contains the definition for the additional Parameters for the Login operation.

Login Parameter

Description	Parameter Name	Value	Type
User Session Behaviour	ACUserSessionCheck	SingleUserSession	String

Example of Source Code in C#.NET

```
UserLoginType userLogin = new UserLoginType();
userLogin.UserName = "Administrator";
userLogin.Password = "eli";
userLogin.Mandant = 7777;
```

```
StringDataType specialLoginParameter = new StringDataType();
specialLoginParameter.Name = "ACUserSessionCheck";
specialLoginParameter.Value = "SingleUserSession";

userLogin.Parameters.StringData = new StringDataType[1] { specialLoginParameter };
```

If the "ACUserSessionCheck" parameter is sent with the Login request, the Login will be rejected and return an error message, if the User Session is already active (i.e. already logged in via WebService). If the User Session is not active a normal Login will occur.

This login feature is useful if multiple webservice clients use the same user to login. The separate WebService processes can determine if the User Session is already active.

LoginType Description

The LoginType object contains information about the user login. The object consists of 2 elements :

1. Login Token – a unique value to identify the current user. The value of the login token is returned by a Login operation and does not change for the current session.
2. User Login – the login informations to identify the user and the Mandant database.

The LoginType object is also required for other operations to identify the current user.

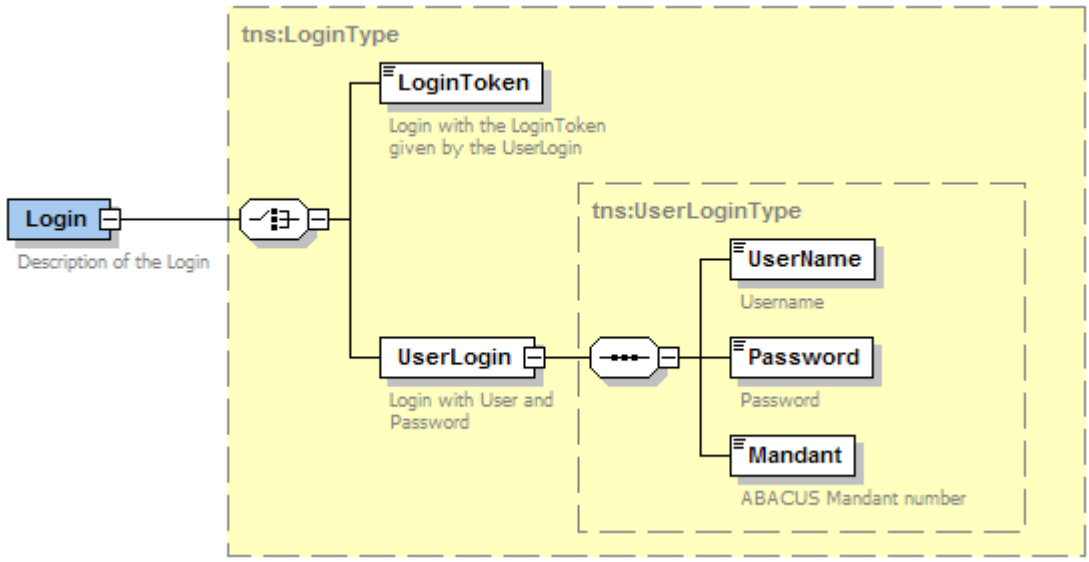

Fig. LoginType Object for AbaConnect WebServices

The Login operation requires the Username, Password and Mandant. The Login operation returns a unique identifier call a “Login Token” that can be used to identify the current user for subsequent operations where the Login information is required.

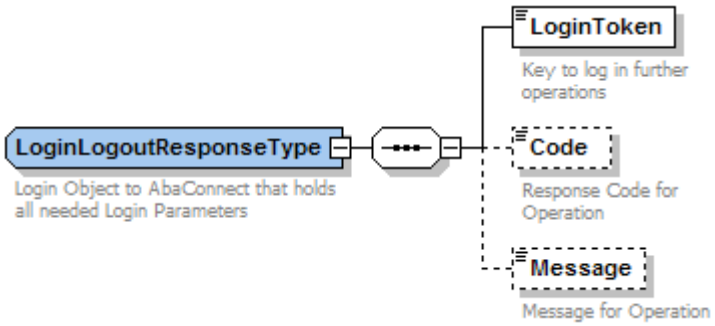

Fig. LoginLogoutResponseType Object for AbaConnect WebServices

Example C#.NET Source Code for Login :

```
// Login with Username, Password and Mandant
UserLoginType userLogin = new UserLoginType();
userLogin.UserName = "Administrator";
userLogin.Password = "eli";
userLogin.Mandant = 7777;

LoginType loginType = new LoginType();
loginType.Item = userLogin;

LoginLogoutResponseType loginResponse = webServicePortObject.login(loginType);
// Get the login token returned from Login
String  currentLoginToken = loginResponse.LoginToken;
```

Example JAVA JAX-WS Source Code for Login :

```
// Login with Username, Password and Mandant
UserLoginType userLogin = new UserLoginType();
userLogin.setUserName("Administrator");
userLogin.setPassword("eli");
userLogin.setMandant(7777);

LoginType loginType = new LoginType();
loginType.setUserLogin(userLogin);

LoginLogoutResponseType loginResponse = webServicePortObject.login(loginType);
// Get the login token returned from Login
String currentLoginToken = loginResponse.getLoginToken();
```

Note : An example of the SOAP Messages that this Source Code might create is shown in the "*WebService SOAP Overview*" section on the previous pages.

Using Login Token or Username, Password and Mandant

Most of the AbaConnect data operations require that the user login information is specified using a LoginType parameter. The LoginType parameter contains either the Login Token or the Username, Password and Mandant information to identify the user. It is recommended to use the “Login Token” because it limits the number of times the username and password are sent over the SOAP connection. The “Login Token” is a unique identifier that changes with every login session. The Username, Password and Mandant are necessary for the initial Login operation. For further data operations, the LoginToken can be used.

Determining If Login Session Is Active

The status of the login session can be checked with the isAlive operation. The isAlive operation requires the “Login Token”. Therefore, the Login Token from the Login response is required.

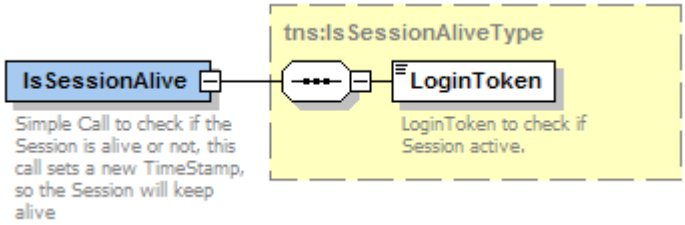

Fig. IsSessionAlive Request Object for AbaConnect WebServices

The isAlive operation will return false if the user session is no longer active.

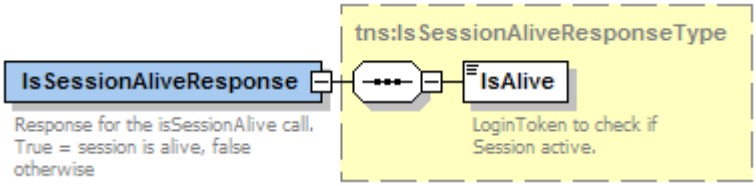

Fig. IsSessionAliveResponse Object for AbaConnect WebServices

Hint : Depending on network configurations, a SOAP connection may be deactivated if too many other connections are open. A call across the SOAP connection should reactivate the connection, if it was closed. The operation “isAlive” can be used, simultaneously, to check if the SOAP connection is still active and the login session is active.

Login To Mandant

For the AbaConnect WebService Login, it is necessary to supply the Mandant number. All following operations executed during the login session, apply to the Mandant specified during login. To change the Mandant it is necessary to logout and login with another Mandant.

Special Notes : User Sessions

It is not possible make multiple logins into ABACUS with a single user. The same principles exists for an ABACUS user login via the AbaMenu as for the AbaConnect WebServices (i.e. one user – one login session). If multiple logins are executed for a user via the AbaConnect interfaces, the current active user session will be adopted for the login.

- The same ABACUS User should not be used to login from more than one parallel process.
- The same ABACUS User should not be used, simultaneously, to login via the ABACUS Menu and AbaConnect WebServices.

In general, it is recommended to use a dedicated ABACUS User for AbaConnect WebService operations (e.g. one ABACUS User should be used for a single process).

AbaConnect WebService Data Operations

The following data operations are available in AbaConnect WebService interfaces :

Import Operations :

- **Insert** – inserts a new data record
- **Update** – updates an existing data record
- **Save** – saves a data record, combination of Insert and Update
- **Delete** – deletes an existing data record

Export Operations :

- **Find** – searches for existing data records
- **DefaultValues** – retrieves a default data structure

These operations are optional for an AbaConnect WebService interface. Depending on the Application interface these operation may be available, or not.

The Operation defines the default "mode" for the data import. The import operations Insert, Update, Save and Delete can be compared to the "mode" attribute used for the import of data via AbaConnect XML files (e.g. mode="SAVE", mode="INSERT", mode="DELETE").

All data operations need to identify the ABACUS User with login information. This can be done via the Login information in the Request. The Login information can contain either the LoginToken from the Login or the Username, Password and Mandant information. It is recommended to use the LoginToken, rather than send the Username and Password for every request. The AbaConnect WebService examples available on the ABACUS internet Homepage provide good examples of how to use the LoginToken.

The DefaultValues operation is optional and may not be implemented by all interfaces. DefaultValues can be used to return a default data structure for a particular interface. The default data structure can then be completed with actual data values and used in a further save requests. Usually, a basic data structure can be generated, without the need to use the DefaultValues operation. DefaultValues can be compared to selecting "New" in the ABACUS Menu to create default "empty" input form for input of a new data record via the ABACUS UI Programs.

Structure of an AbaConnect Data Request

The structure of an AbaConnect data request is the same for all data operations. The 4 main elements of a Request are :

- **AbaConnectParam** – defines the Login information, Revision number, Response message Level and Additional Parameters
- **ApplicationParam** – defines application specific parameters that may required for import and export data operations. The application parameters that are available for a particular WebService interface are generally documented on the ABACUS Homepage (see AbaConnect WebServices Downloads).
- **FindParam** – defines the Find parameters used to search and export data with the Find operation. It is not used for Save, Insert, Update and Delete operations. The application parameters that are available for a particular WebService interface are generally documented on the

ABACUS Homepage (see AbaConnect WebServices Downloads).

- **Data** – defines the data structure for import operation for Save, Insert, Update and Delete. It is not used for Find operations.

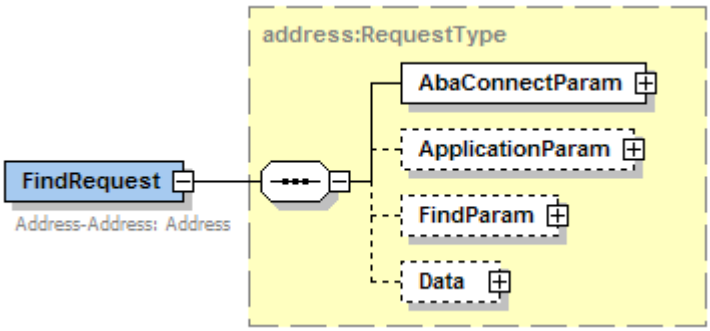

Fig. Structure of Data Request Object for AbaConnect WebServices

The Figure above shows the structure for a FindRequest. The the Request structure is identical for Save, Insert, Update and Delete. All data requests via AbaConnect WebServices must include the AbaConnectParam for the definition of the User information. Depending on the Request that is sent, it may be required to send the ApplicationParam. FindParam and Data elements.

The Request structure viewed as a pure SOAP Envelope structure has the structure shown below. Depending on the request operation that is called, the “FindRequest” element could be “SaveRequest”, “InsertRequest”, “UpdateRequest” or “DeleteRequest”.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header></SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <ns1:FindRequest xmlns:ns1="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
         <ns1:AbaConnectParam>
            <ns1:Login>
               <ns1:LoginToken>f9f09cf6657034434eff61055062ff8d18fd637d</ns1:LoginToken>
            </ns1:Login>
            <ns1:Revision>0</ns1:Revision>
            <ns1:Level>Warning</ns1:Level>
            <ns1:Additional>
               …... (optional contains additional parameters)
            </ns1:Additional>
         </ns1:AbaConnectParam>
         <ns1:ApplicationParam>
               …... (optional contains application specific parameters)
         </ns1:ApplicationParam>
         <ns1:FindParam>
            <ns1:Index>1</ns1:Index>
            <ns1:Operation>GREATER_EQUAL</ns1:Operation>
            <ns1:KeyFields>
               …. (this contains an application specific key fields.  Normally required for a Find Operation for specific data record)
            </ns1:KeyFields>
         </ns1:FindParam>
        <ns1:Data>
            …. (this contains an application specific data structure. Normally required for a Save operation)
         </ns1:Data>
      </ns1:FindRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

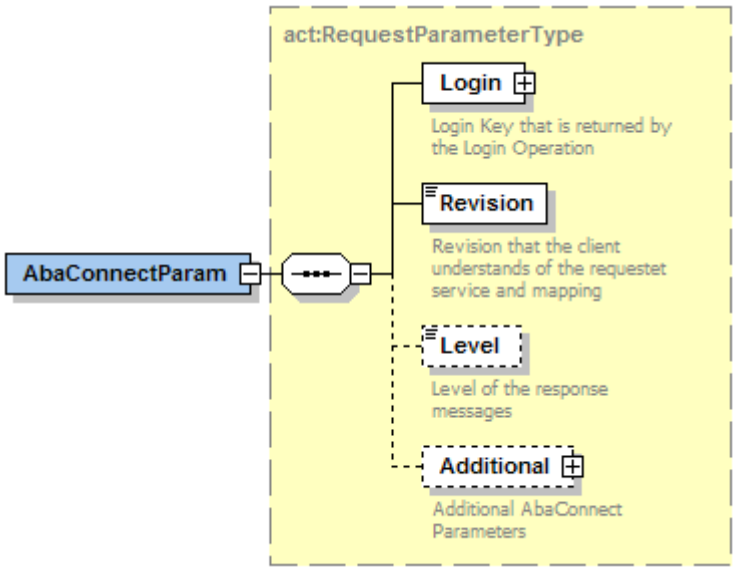

Fig. AbaConnectParam Object Structure for AbaConnect WebServices

The 4 main elements of the AbaConnectParam object are :

- **Login** – the user login information or a login token
- **Revision** – the Revision number of the interface (default is 0), In most cases this can be zero unless a specific revision number for an interface exists.
- **Level** – the returned message level. There are 3 message levels : INFO, WARNING or ERROR. For most applications the WARNING level is used. The INFO level returns the most information and the ERROR level returns the least.
- **Additional** – additional parameters that can be supplied via the interfaces, e.g. request for ExtendedFields or Dossiers. “ACIncludeExtendedFields”, “ACGAPDossiersInclude”

The AbaConnect WebService examples available on the ABACUS internet Homepage provide good examples of how the AbaConnectParam is used.

The Revision number defines which data fields for an interface are returned via WebService. In general, the Revision number can be set to 0. When an AbaConnect interface is first released, all the available XML data fields have a revision number of 0. If fields are added to the AbaConnect interface after its release, the additional fields will be given a revision number 1 higher than the last revision (i.e. 1, 2, 3, etc.). As a rule, a revision number 0 should be used, unless particular XML data fields are required with a revision number greater than 0. If a revision number greater than 0 is used, it should be confirmed that the WSDL Schema being used contains the required XML field definitions for the request Revision. The specific Revision number for the XML data fields can be viewed in the AbaConnect documentation in the ABACUS Homepage or via ABACUS UI Prog 625 AbaConnect Configuration. The main influence of the Revison number is for the Find Request. The Revision number will normally be ignored during Save, Insert and Update Request, as the data structure that is sent defines what fields are to be saved.

Application Parameters (ApplicationParam)

Application Parameters are part of the AbaConnect WebService Request structure. Application Parameters can be used when retrieving or saving data (e.g. FindRequest, SaveRequest, etc). Application Parameters are related to a particular application

interface.

Examples of Application Parameters:

Application	Interface	Request Type	KeyFields
Ordering ABEA (ORDE)	ordesalesorder_20xx_00	SaveRequest	NextStatus (Bool)
Ordering ABEA (ORDE)	ordesalesorder_20xx_00	SaveRequest	SetPriceFinding (Int)
Account Payable (KRED)	kreddocument_20xx_00	FindRequest	DocumentWithoutPicture (Bool)
Payroll (LOHN)	HierarchyEmployee_20xx_00	FindRequest	lohnsst_date_per (Date)
Payroll (LOHN)	HierarchyEmployee_20xx_00	FindRequest	lohnsst_date_bis (Date)

Further Application Parameters are documented on the ABACUS Homepage. At the end of each Application section of the AbaConnect WebServices, is a document "WebService Parameters". The "WebService Parameters" document contains possible Application Parameters for specific AbaConnect interfaces.

Please consult the "WebService Parameters" documentation on the ABACUS Homepage to check the available Application Parameters for specific AbaConnect interfaces. The AbaConnect WebService examples available on the ABACUS internet Homepage also provide examples of how the Application Parameters are used.

The Application Parameters are named and typed Parameters of an ObjectDataType. The ObjectDataType can contain the following data types :

- BooleanDataType – boolean type
- StringDataType – string type
- IntDataType – integer type
- LongDataType – long type
- DecimalDataType – decimal type
- DateDataType – data type
- DateTimeDataType – data and time type

Application Parameters typically correlate to the options that are available via the ABACUS UI, when importing or exporting data via XML-files. Many of the Application Parameters available for AbaConnect WebServices can be directly related to parameters available in ABACUS UI Program 625.

Additional Parameters

Additional Parameters are are central parameters, typically sent with a Find Request, when retrieving data for a particular interface. The Additional Parameters are not generally specific to a particular AbaConnect interface, but the interface must support the functionality. For example, if an Application interface supports Extended Fields or Dossiers the related Additional parameters can be supplied with a request to obtain the Extended Fields information or Dossier object linked to the data records. The information will only be returned where the Application interface supports these features.

Description	Parameter Name	Value	Type	Example
Extended Fields	ACIncludeExtendedFields	all	String	ACIncludeExtendedFields = “all”
Dossiers	ACGAPDossiersInclude	TRUE	Boolean	ACGAPDossiersInclude = true

Empty Fields	ACIncludeEmptyElements	TRUE	Boolean	ACIncludeEmptyElements = true
Multiple Data Records (ab Abacus V2021)	ACDataRecordCount	1 - 50	Integer	ACDataRecordCount = 10 (ab Abacus V2021)

Extended Fields – if an interface supports extended Fields the Extended Fields can be retrieved with the “ACIncludeExtendedFields” parameter. Via WebService, the Extended Fields are returned as a list in a dynamic array (list of “ObjectDataType”). Each list element contains a name and a value. The name of the Extended Field is the internal ABACUS field name for the Extended Fields (e.g. _USERFIELD1, _USERFIELD2, etc.). The placement and naming of the Extended Fields in the data structure is identical to the way Extended Fields are transferred using the ABACUS Menu Program 625 when Exporting and Importing AbaConnect XML-files via the ABACUS UI. The only difference with the WebService is that the Extended Fields are represented in a dynamic array list, instead of directly part of the XML structure itself. The "ACIncludeExtendedFields" is a String type parameter and the only possible value is "all". Please refer to the detailed notes in this document "*Handling of Extended Fields in AbaConnect WebServices*" for more information about handling ExtendedFields via the AbaConnect interfaces.

Dossiers – if an AbaConnect interface implements a Dossier Element, the associated binary data can be exported and imported also via the WebService interface.

Note on Dossiers : Generally, the application interfaces that contain the Dossier Element, return Dossiers by default if the “ACGAPDossiersInclude” parameter is not present in the Find-Request. Therefore, it is recommended to always set the “ACGAPDossiersInclude” flag in the Find-Request with either “true” or “false” if the interface supports Dossiers as a sub element. If Dossiers are not required then set the “ACGAPDossiersInclude” parameter to “false”, to omit the Dossier data and reduce the size of the Response. When using the pure Dossier interfaces such as CentralDossier_2010_00 and DocumentDossierAccountsReceivable_2009.00, etc, the .“ACGAPDossiersInclude” parameter should be set to “true”.

Empty Fields – by default empty fields (e.g. Strings) are omitted from the returned Structure with WebService Find Requests. To override this feature, the Additional parameter “ACIncludeEmptyElements” = true can be sent with the Find Request. This may be useful when working with ExtendedFields, to retrieve the names of empty ExtendedField.

Multiple Data Records – As of Abacus V2021 it is possible to export multiple data records with one request when navigating the data records using Find-Request - FIRST, NEXT, LAST and PRIOR operations (see further information for *Multiple Data Records* on page 35).

Find Request Operation

A Find Request is used to search and return specific data for an interface. The AbaConnect WebService examples available on the ABACUS internet Homepage show how some of the parameters may be used to retrieve data, and also show the use of the Additional Parameters for Extended Fields and Dossier objects for a data record.

The Find Parameters for a particular interface are generally closely related to the database indexes for the main database table referenced by the AbaConnect interface. Please note, that there are some interfaces available that do not reference database tables, and in some cases not all indexes available in a database table, are available for use via the interface. The first place of reference for the available interface indexes and related Find Parameters is the "WebService Parameters" documentation.

The available Find Parameters for a particular AbaConnect interface are documented on the ABACUS Homepage. At the end of each Application section of the AbaConnect WebServices, is a document "WebService Parameters". The "WebService Parameters" document contains possible Find Parameters for specific AbaConnect interfaces.

The AbaConnect WebService examples available on the ABACUS internet Homepage also provide examples of how the Find Parameters are used.

A typical Find Request for LOHN HierarchyEmployee interface would have the following structure when viewing the raw SOAP Envelope data sent via the WebService.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header></SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <ns1:FindRequest xmlns:ns1="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
         <ns1:AbaConnectParam>
            <ns1:Login>
               <ns1:LoginToken>f9f09cf6657034434eff61055062ff8d18fd637d</ns1:LoginToken>
            </ns1:Login>
            <ns1:Revision>0</ns1:Revision>
            <ns1:Level>Warning</ns1:Level>
            <ns1:Additional>
               <ns1:StringData Name="ACIncludeExtendedFields">all</ns1:StringData>
               <ns1:BooleanData Name="ACGAPDossiersInclude">false</ns1:BooleanData>
            </ns1:Additional>
         </ns1:AbaConnectParam>
         <ns1:FindParam>
            <ns1:Index>1</ns1:Index>
            <ns1:Operation>GREATER_EQUAL</ns1:Operation>
            <ns1:KeyFields>
               <ns1:LongData Name="EmployeeNumber">2</ns1:LongData>
            </ns1:KeyFields>
         </ns1:FindParam>
      </ns1:FindRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Structure of a FindParam

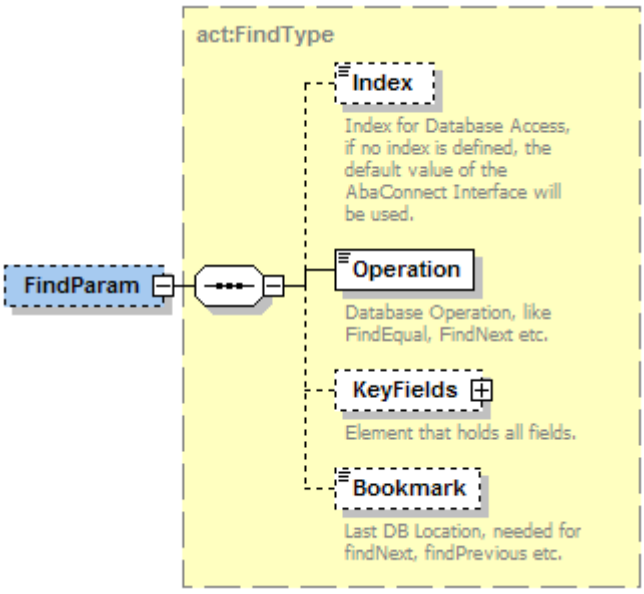

Fig. : Structure of the FindParam object

The FindParam object has 4 parameters :

- **Index** – used to specify the search index. The Index is closely linked to the database index for the main ABACUS database table of the interface. The index defines the sorting order of the data records and also the search parameters that are available for the specified index.
- **Operation** – used to specify the Find operation. The possible Find operations are documented below.
- **KeyFields** – used to specify search parameters
- **Bookmark** – used to specify a Bookmark from a known data record

The Index of the Find Parameter is closely linked to the index of the the ABACUS database table. The Index defines the data-segments available for a particular AbaConnect WebService interface. The sorting order of the data records is also defined by the Index (i.e. defined by the data segment fields for the available index).

The following OperationType are available for AbaConnect WebService interfaces

Operation Type	Description
FIRST	Locates the first data record according to the specified index. No additional parameters, other than the index, are necessary for Find FIRST. This is typically used for browsing records with NEXT and PRIOR without using additional search parameters.
LAST	Locates the last data record for the specified index. No additional parameters, other than the index, are necessary for Find LAST. This is typically used for browsing records with PRIOR and NEXT without using additional search parameters.
NEXT	Locates the next data record from a known data record for the specified index. The known data record position can be specified by a Bookmark. A Bookmark can be read from a known data record.
PRIOR	Locates the previous data record from a known data record for the

Operation Type	Description
	specified index. The known data record position can be specified by a Bookmark. A Bookmark can be read from a known data record.
EQUAL	Locates a data record where the values for a specified index exactly match the specified parameters. The Find.EQUAL can also be used with a Bookmark (e.g. After a SaveRequest a Bookmark is returned, which can be used with Find.EQUAL with the Bookark to locate the saved data) If the parameters do not uniquely specify a data record then Find EQUAL may not return any result.
GREATER_EQUAL	Locates the first data record where the values for a specified index are greater than or equal to the specified parameters. This is typically used to browse records in ascending order of the index.
LESS_EQUAL	Locates the last data record where the values for a specified index are less than or equal to the specified parameters.
GREATER	Locates the first data record where the values for a specified index are greater than the specified parameters. This is typically used to jump to the first detail record of a master-detail relationship.
LESS	Locates the last data record where the values for a specified index are less than or equal to the specified parameters. This is typically used to browse records in reverse order of the index.

Note : Bookmarks should not be used to identify data records outside a single login session. When the database is updated, the bookmarks can change, so they should not be used externally to identify records for more than a single login session. The best way to identify data is via the unique identifies for the specific Application data records. The unique identifiers are usually associated with a Find Index.

Examples of Find Parameters:

Application	**Interface**	**Index**	**KeyFields**
Address Management	Address_20xx_00	1	AddressNumber (Long)
Account Receivable (DEBI)	Customer_20xx_00	1	CustomerNumber (Long)
Account Payable (KRED)	kredsupplier_20xx_00	1	SupplierNumber (Long)
Payroll (LOHN)	HierarchyEmployee_20xx_00	1	EmployeeNumber (Long)
Project Management (PROJ)	ProjectBase_20xx_00	1	ProjectNumber (Long)

Further Find Parameters are documented on the ABACUS Homepage. At the end of each Application section of the AbaConnect WebServices, is a document "WebService Parameters". The "WebService Parameters" document contains the possible Find Parameters for specific AbaConnect interfaces.

KeyFields define the search parameters that can be used for a particular interface. The KeyFields are application-specific. The KeyFields are usually closely linked to the ABACUS Database Table Indexes. The WebService FindParams may or may not include

indexes from the ABACUS database tables. Please consult the WebService Find Parameters documentation to check the available KeyFields.

The AbaConnect WebService examples available on the ABACUS internet Homepage provide good examples of how the KeyFields are used. The available KeyFields for a particular WebService interface are documented on the ABACUS Homepage (see AbaConnect WebServices Downloads).

The KeyFields are named and typed Parameters of an ObjectDataType. The ObjectDataType can contain the following data types :

- BooleanDataType – boolean type
- StringDataType – string type
- IntDataType – integer type
- LongDataType – long type
- DecimalDataType – decimal type
- DateDataType – data type
- DateTimeDataType – data and time type

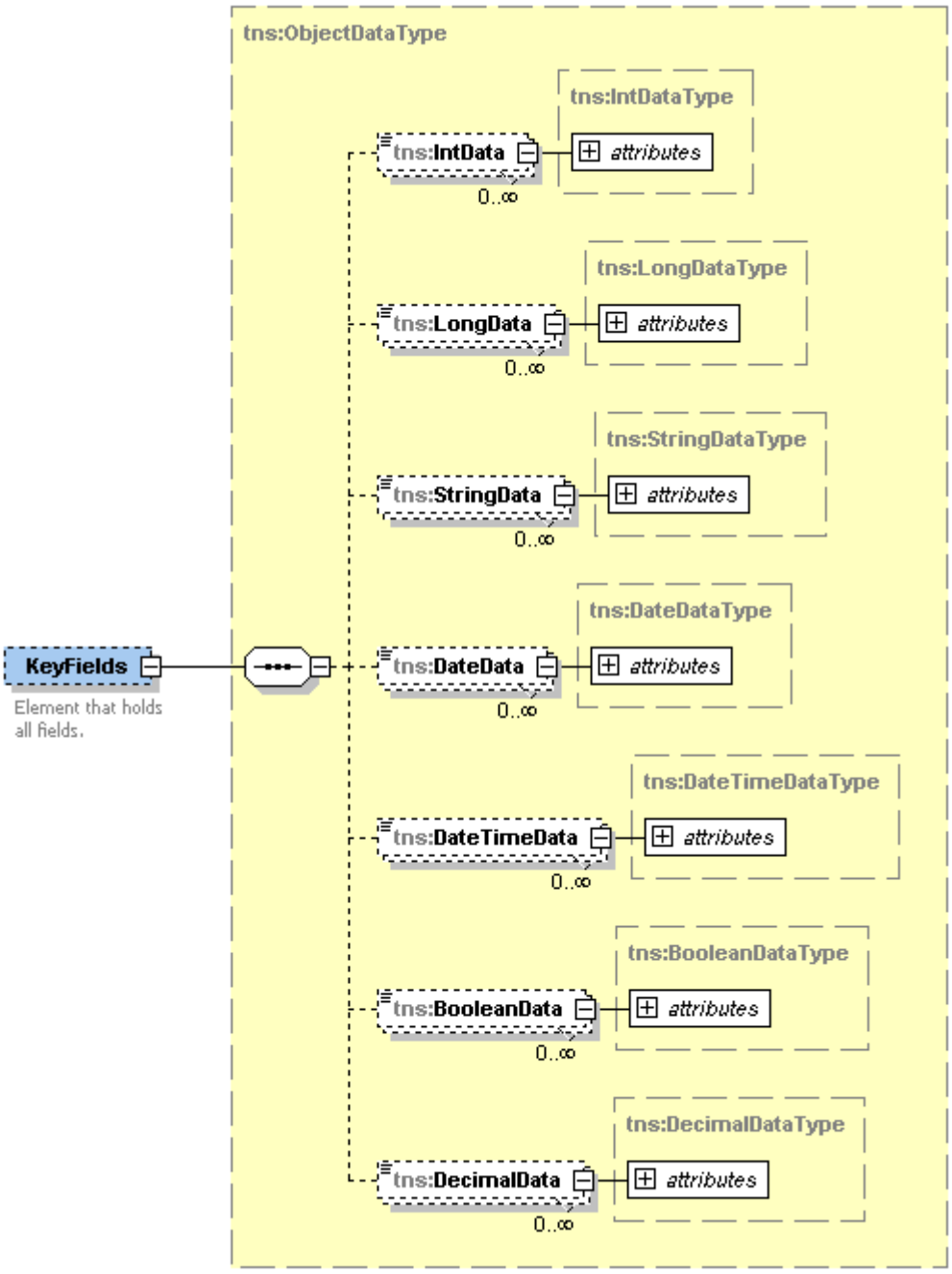

Fig. : Structure of the KeyFields object showing ObjectDataType possibilities

Find Index Defines the Sorting Order of Records

When a Find operation is called the Find Index defines the sorting order of the records and also the specific parameters that can be used for the Find Operation to request data. Generally, the Find.FIRST followed by Find.NEXT can be used to browse through data in the order defined by the Index.

Usage of Find FIRST, NEXT, LAST, PRIOR

The Find Operations of FIRST, NEXT, LAST and PRIOR are the most common ways to browse through data records. Find FIRST and LAST navigate to the first and last record, respectively, in the order defined by the Index. The NEXT and PRIOR require then the Bookmark of the current record, and navigate to the next and prior record, respectively, in

the order defined by the Index.

Usage of Find EQUAL and GREATER_EQUAL and KeyFields
The Find Operation with EQUAL or GREATER_EQUAL is the most common way to request specific data records combined with Find "KeyFields" parameters

Usage of Find EQUAL and Bookmark after Save Operation
The Find Operation with EQUAL can be used together with the Bookmark. The most common use for this, is after a Save Request. The SaveRequest will return a Bookmark when the data is successfully saved. The Bookmark is a unique reference for the record. After a Save Operation a Find EQUAL with Bookmark can be used to retrieve the data that was saved. This is useful to retrieve the ABACUS specific Identifiers for the data record if it was not specified in the Save Request. For example, an Address could be saved without defining the AddressNumber, and after the SaveRequest, the saved record could be retrieved with Find EQUAL with Bookmark. The AddressNumber can then be read from the returned data structure.

Save, Insert und Update Request Operation

The Save, Insert and Update methods are all import methods to save data in ABACUS. In general, most of the AbaConnect interfaces do not require any additional parameters to be sent other than the Data structure. The Save, Insert and Update methods can all be compared to importing an XML file via the ABACUS Menu in Program 625.

The Operation defines the default "mode" for the data import. The import operations Insert, Update, Save and Delete can be compared to the "mode" attribute used for the import of data via AbaConnect XML files (e.g. mode="SAVE", mode="INSERT", mode="DELETE").

The error messages that occur with a Save, Insert and Update Request can generally be reproduced via an AbaConnect XML Import file via the ABACUS Menu in Program 625 with the same data structure. For debugging and Support possibilities please refer to the usage of the Apache TCPMon Utlility. If the Data structure can be extracted via the WebService SOAP message using Apache TCPMon and converted to an AbaConnect Import file, it is much easier to reproduce the error messages via the ABACUS Menu via Program 625.

The structure of the Save, Insert and Update methods contains the data fields from the interface Structure. It is the same XML structure that is used in an XML Import file. Generally, the import is possible without any further parameters. In some cases such as the Finance Bookkeeping interfaces additional Application Parameters can be used to define the booking year (FibuJournal). The application parameters are defined in the documentation available in the ABACUS Homepage. The AbaConnect WebService examples also provide important information about how the parameters can be used with WebServices.

The following WebService Operations are available :

Import Operations :

- **Insert** – inserts a new data record

- **Update** – updates an existing data record
- **Save** – saves a data record, combination of Insert and Update
- **Delete** – deletes an existing data record

All of the above operations (including the Delete operation) transfer an XML structure to the server via WebService. The Delete operation is documented separately, although it is also an import method. The Delete operation is identical to the Save operation.

An example WebService operation for a Save Request may look like the following for a Save Request :

```
<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Body>
   <ns2:SaveRequest xmlns:xmime="http://www.w3.org/2005/05/xmlmime"
xmlns:ns3="http://www.abacus.ch/abaconnect/2010.00/adre/AddressTypes"
xmlns:ns2="http://www.abacus.ch/abaconnect/2010.00/adre/Address"
xmlns="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
      <ns2:AbaConnectParam>
         <Login>
            <LoginToken>5cddd67daaa513d24a00d04e1541c723f77df9c1</LoginToken>
         </Login>
         <Revision>0</Revision>
         <Level>Warning</Level>
      </ns2:AbaConnectParam>
      <ns2:Data>
         <ns3:AddressData>
            <ns3:LastName>Smith</ns3:LastName>
            <ns3:FirstName>John</ns3:FirstName>
            <ns3:Country>CH</ns3:Country>
            <ns3:ZIP>9000</ns3:ZIP>
            <ns3:City>St. Gallen</ns3:City>
            <ns3:Language>D</ns3:Language>
            <ns3:Line1>Bahnhofstrasse 234</ns3:Line1>
            <ns3:TextField3>AC_06.01.2014 13.39:38</ns3:TextField3>
         </ns3:AddressData>
      </ns2:Data>
   </ns2:SaveRequest>
</S:Body>
</S:Envelope>
```

The above structure does not contain an “AddressNumber” or CodeName field, so a new Address would be created in ABACUS. For an InsertRequest the stucture would be identical, except that the Request would be and InsertRequest instead of a SaveRequest.

```
<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Body>
   <ns2:InsertRequest xmlns:xmime="http://www.w3.org/2005/05/xmlmime"
xmlns:ns3="http://www.abacus.ch/abaconnect/2010.00/adre/AddressTypes"
xmlns:ns2="http://www.abacus.ch/abaconnect/2010.00/adre/Address"
xmlns="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
      <ns2:AbaConnectParam>
         <Login>
            <LoginToken>5cddd67daaa513d24a00d04e1541c723f77df9c1</LoginToken>
         </Login>
         <Revision>0</Revision>
         <Level>Warning</Level>
      </ns2:AbaConnectParam>
      <ns2:Data>
         <ns3:AddressData>
            <ns3:LastName>Smith</ns3:LastName>
            <ns3:FirstName>John</ns3:FirstName>
            <ns3:Country>CH</ns3:Country>
            <ns3:ZIP>9000</ns3:ZIP>
            <ns3:City>St. Gallen</ns3:City>
            <ns3:Language>D</ns3:Language>
```

```
            <ns3:Line1>Bahnhofstrasse 234</ns3:Line1>
            <ns3:TextField3>AC_06.01.2014 13.39:38</ns3:TextField3>
         </ns3:AddressData>
      </ns2:Data>
   </ns2:InsertRequest>
</S:Body>
</S:Envelope>
```

For and UpdateRequest it is necessary to identify an existing record. This is done by the addition of the main fields for the interface. This is normally a number or a short name. The main fields for an interface are generally related to the main index fields for an interface and are application specific. If it is not clear what are the main fields for an interface, please contact the application support.

```
<?xml version="1.0" ?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Body>
   <ns2:UpdateRequest xmlns:xmime="http://www.w3.org/2005/05/xmlmime"
xmlns:ns3="http://www.abacus.ch/abaconnect/2010.00/adre/AddressTypes"
xmlns:ns2="http://www.abacus.ch/abaconnect/2010.00/adre/Address"
xmlns="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
      <ns2:AbaConnectParam>
         <Login>
            <LoginToken>5cddd67daaa513d24a00d04e1541c723f77df9c1</LoginToken>
         </Login>
         <Revision>0</Revision>
         <Level>Warning</Level>
      </ns2:AbaConnectParam>
      <ns2:Data>
         <ns3:AddressData>
            <ns3:AddressNumber>740147097</ns3:AddressNumber>
            <ns3:LastName>Smith</ns3:LastName>
            <ns3:FirstName>John</ns3:FirstName>
            <ns3:Country>CH</ns3:Country>
            <ns3:ZIP>9000</ns3:ZIP>
            <ns3:City>St. Gallen</ns3:City>
            <ns3:Language>D</ns3:Language>
            <ns3:Line1>Bahnhofstrasse 456</ns3:Line1>
            <ns3:TextField3>AC_06.01.2014 14.08:52</ns3:TextField3>
         </ns3:AddressData>
      </ns2:Data>
   </ns2:UpdateRequest>
</S:Body>
</S:Envelope>
```

An update (modification) for an existing record can occur via an UpdateRequest or a SaveRequest. If the main fields of an interface are supplied and identify an existing record, then a SaveRequest will update the existing record. Otherwise a new record will be created. If an existing record is identified for a SaveRequest, then the existing record will be updated. Otherwise a new record will be created. The SaveRequest is a combination of the Insert and Update operation. Generally, if the update and inserts need to be exactly controlled from the client WebService application then the UpdateRequest and InsertRequest can be used. Otherwise, the SaveRequest can be used.

If an interface only allows new records to be created then only the InsertRequest will be available. Most interfaces allow both modifications and new records to be created and all 3 Request possibilities are available.

Note : Some of the ABEA interfaces only implement the SaveRequest. Via the SaveRequest, an Insert or Update will be executed. If the data structure identifies an existing record via the main fields, then an existing record will be updated, or otherwise a new record will be created.

Delete Request Operation

The Delete Request to delete a record in identical to the Save Request. The data structure to delete is sent via the interface. The data structure must contain a minimum number of data fields to identify the record. The minimum fields required for the delete is defined by the application.

```
<?xml version="1.0" ?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Body>
   <ns2:DeleteRequest xmlns:xmime="http://www.w3.org/2005/05/xmlmime"
xmlns:ns3="http://www.abacus.ch/abaconnect/2009.00/adre/AddressTypes"
xmlns:ns2="http://www.abacus.ch/abaconnect/2009.00/adre/Address"
xmlns="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
      <ns2:AbaConnectParam>
         <Login>
            <LoginToken>0d05bf6f5e8f909c107ba729f209fe862dfb827b</LoginToken>
         </Login>
         <Revision>1</Revision>
         <Level>Warning</Level>
      </ns2:AbaConnectParam>
      <ns2:Data>
         <ns3:AddressData>
            <ns3:AddressNumber>740147090</ns3:AddressNumber>
            <ns3:Line1>Hauptstr. 50</ns3:Line1>
            <ns3:LastName>Smith</ns3:LastName>
            <ns3:FirstName>John</ns3:FirstName>
            <ns3:Country>CH</ns3:Country>
            <ns3:ZIP>9001</ns3:ZIP>
            <ns3:City>St. Gallen</ns3:City>
            <ns3:Language>D</ns3:Language>
         </ns3:AddressData>
      </ns2:Data>
   </ns2:DeleteRequest>
</S:Body>
</S:Envelope>
```

Procedures for IsFinished Request

All AbaConnect WebService requests which import or export data (i.e. find, save, insert, update, delete and defaultValues), require an implementation for "isFinished" which follows the initial request. For example, a Find-Request or Save-Request will return a Response with a boolean parameter value "IsFinished". If this value "IsFinished" = false, then the request must be further polled with the "IsFInishedRequest", with the corresponding "RequestID", until the request returns with "IsFinished" = true

The IsFinished polling procedure represents the data processing interval on the Abacus Server. Many data Requests may return "IsFinished" = true, directly after the initial requests. Other requests may need one or more loops with IsFinished, before the data processing on the server is completed.

If a data request returns IsFinished = "false", then it is necessary to complete the request with the "IsFinishedRequest" processing loop. Examples of this procedure are demonstrated in the AbaConnect WebService examples available on the Abacus Homepage.

It is usual for the first request to take longer, due the the resources that must be loaded on

the server. Following requests to the same interface will be quicker. The time taken for a Response is dependent on the interface, the data size, the load on the Abacus server, and the network connection. A general indication of a request time for an AbaConnect data request is between 30 and 100 milliseconds

The use of the "IsFinishedRequest" keeps the request and response time short, and helps prevents associated problems due to network port connection lifetimes and timeouts.

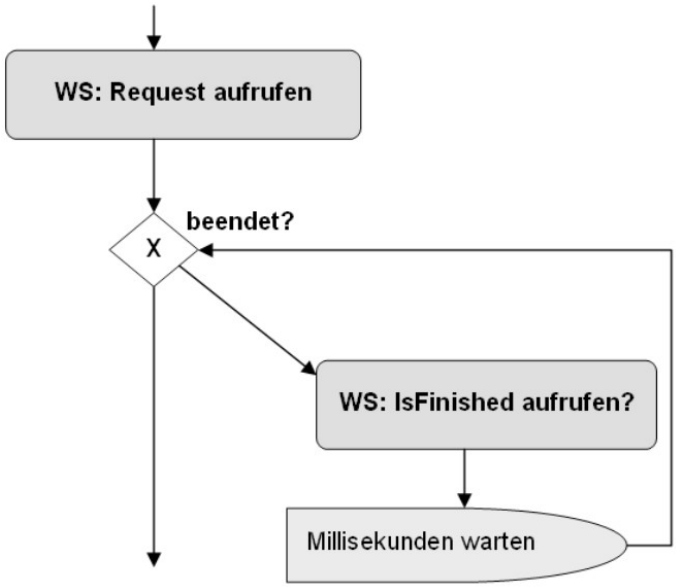

Fig. Representation of a Data-Request Process for IsFinished
After the initial request the "IsFinished" Response parameter muss be checked. If "IsFinished" = true, the data processing on the Abacus Server is completed and the Client WebService program flow can continue without the IsFinished polling. If the initial request returns "IsFinished" = false, the corresponding Response parameter "RequestID" will contain a value, and the "IsFinishedRequest" must be called until the Response returns with "IsFinished" = true.

The "IsFinished" loop should contain a small wait/sleep between the IsFinished calls. It is recommended to implement a wait/sleep of 100 to 250 milliseconds between IsFinishedRequests.

Note : A data Request remains pending until the Request returns with “IsFinished” = true. A pending request will block other requests in the same user session, until the RequestID is cleared with the correct “IsFinished” procedure.

The occurance of the following error message (8040) :

There is already a Request Pending with ID[[RA1FEL5JBJAM4SFO4XTY2PROQF]]. Please wait until it's finished

during data requests, indicates that at least one previous request for the same user session has not been correctly completed with the IsFinished Requests. The RequestID output with the message, could be used to identify the data Request that is not completed, or possibly was not correctly completed.

Note : It is important to properly handle Exceptions during the “IsFinished” polling loop, so

that conditions can be handled on the client side, where the polling loop has prematurely ended, without fully completing the “IsFinished” polling.

The "IsFinished" response of "true", indicates that the data processing is ***finished***. It does not indicate that the data processing was successful. All data requests must return a status "IsFinished" = true. To determine if errors or warnings occurred, the Response messages must be analysed. Generally, the presence of error messages indicates that data was not successfully saved.

Handling Exceptions During AbaConnect Requests and Responses

The AbaConnect Interface implement exceptions and it is very important to handle these exceptions during the request and response handling. Exceptions can be thrown from the ABACUS Server if something unexpected occurs, or it can be thrown due to communication errors (e.g breakdown in the network connection).

Note : If Exceptions are not correctly handled by the client WebService Program it may create invalid states on the ABACUS Server, that can interrupt the operation of the Client WebService program.

It is particularly important to handle the Exceptions during the data requests where the IsFinished operation is required. If there is a connection problem during the IsFinished polling, and the IsFinished is not polled until the Request returns with “IsFinished” = “true”, then the current “RequestID” may not be cleared on the ABACUS Server. This will block further request to the ABACUS Server, until the current RequestID is cleared.

Important : The only correct way to clear the current RequestID on the ABACUS Server is to ensure the “IsFinished” requests run until “IsFinished” = “true”.

If the connection is interrupted during the Request process, and the ABACUS Server is blocking further Requests due to a remaining RequestID, the RequestID can still be cleared when the connection is re-established, with a further IsFinished Request with the same RequestID.

As a last resort, a Logout will close the actual User Session on the ABACUS Server and clear any remaining RequestID. A new Login Session will then allow requests to be sent again. It should be mentioned, that any running processes in the User Session will also be forced to close, which may contribute to System instability. Also, the status of the pending Request will be lost (i.e. whether the Request was successful, or not), when the User Session is closed.

If messages, such as the following, are returned as Messages via AbaConnect interface Requests or are present in the ABACUS Log files, then it means that the AbaConnect WebService Client program has not correctly handled the Request calls with IsFinished.

```
[WARNING]: There is already a Request Pending with ID[[99EXUPAGOTVAX01XAH]]. (8040)
```

The AbaConnect WebService Examples on the ABACUS Homepage also contain information about how to handle the Exceptions in .NET and JAVA.

With a WebService generally, several things due to exceptions and error handling should be handled by the Client WebService program :

1. No connection to the server - that means that the communication is not available (or incorrectly configured). This will normally result in an Exception on the Client-side e.g. a Connection Timeout or Bad Gateway, etc. The Client Program should catch these exceptions and respond accordingly.
2. Abacus configuration, or internal services are not reachable. This will generally also lead to an Exception. e.g. will a HTTP Code 500, Bad Gateway, etc.
3. Unexpected internal Abacus errors that are not caught by the application (e.g. NullPointerExceptions, DataBase access problems, etc.). These errors are not frequent, but can occur. They will generally be returned as an "AbaConnectFault" which is a form of "SoapFault". This is a an Exception which is returned via SOAP and should also be handled by the Client Program. As SoapFault are also Exceptions, they can be caught by the Client Program. Depending on if the Exception type (i.e. AbaConnectFault or other Exception), it is generally clear whether the exception falls into the above category 1, 2 or 3.
4. Error messages via AbaConnect Responses. These are error messages generated from AbaConnect Framework or from Application-specific errors that occur when saving data e.g. Account not found in Abacus, etc. Some of the general error codes that can be returned from AbaConnect Framework are listed on the Homepage under "AbaConnect General Error Codes" :
   https://downloads.abacus.ch/downloads-page/abaconnect/webservices

Realistically, these exceptions and AbaConnectFault errors should not occur frequently. It is more common that Application-specific errors occur when the data is validated during the save routines.

Generally, if an application error occurs the data should not be saved. The error messages are available in the WebService response message. If no errors occur (i.e. error count = 0), then the data should be saved. The application errors that occur are exactly the same as those that would be returned in a Response file, when the same data-structure would be imported via the AbaConnect XML file import in the Abacus Menu.

Handling of Extended Fields in AbaConnect WebServices

An AbaConnect Interface may or may not support Extended Fields. If Extended Fields are supported in an AbaConnect interface, they can be exported and imported via the interface. Because Extended Fields are User-defined Fields they are dependent on the particular ABACUS Mandant. Therefore, they are added dynamically to an Interface stucture depending on the Mandant.

1. Extended Fields are always added at the end of a data structure
2. There can be more than 1 Element structure in an interface that contains Extended Fields
3. ExtendedFields always use the internal name of the field (e.g. _USERFIELDxx, where "xx" is the number of the UserField)
4. The data structure for export is identical to the data structure for import.

Although the handling of ExtendedFields is similar for AbaConnect XML Files and AbaConnect WebServices. there are some small differences. In AbaConnect XML Files ExtendedFields are added directly to the end of a data-structure, as addition fields, using the internal name of the Extended Field :

Example : XML File Structure for Address Interface :

```
<AddressData mode="SAVE">
        <AddressNumber>75</AddressNumber>
        <Country>CH</Country>
        <ZIP>8000</ZIP>
        <City>Zürich</City>
        <Language>D</Language>
        <PostRoute>0</PostRoute>
        ....
        ....
        <SubjectType>1</SubjectType>
        <AANMainSubject>0</AANMainSubject>
        <AddressValidAsOf>2000-01-01</AddressValidAsOf>
        <PostalCodeSupplement>0</PostalCodeSupplement>
        <_USERFIELD7>2004-01-10</_USERFIELD7>
        <_USERFIELD1>false</_USERFIELD1>
        <_USERFIELD4>false</_USERFIELD4>
        <_USERFIELD5>true</_USERFIELD5>
        <_USERFIELD6>true</_USERFIELD6>
        <_USERFIELD9>false</_USERFIELD9>
</AddressData>
```

In an XML File the Extended Fields are added directly at the end of a data structure. The name of the field is used directly as the XML Element name.

Example : SOAP Structure for Address Interface :

```
<apt:AddressData>
        <apt:AddressNumber>75</apt:AddressNumber>
        <apt:Country>CH</apt:Country>
        <apt:ZIP>8000</apt:ZIP>
        <apt:City>Zürich</apt:City>
        <apt:Language>D</apt:Language>
        <apt:PostRoute>0</apt:PostRoute>
        ....
        ....
        <apt:SubjectType>1</apt:SubjectType>
        <apt:AANMainSubject>0</apt:AANMainSubject>
        <apt:AddressValidAsOf>2000-01-01</apt:AddressValidAsOf>
        <apt:PostalCodeSupplement>0</apt:PostalCodeSupplement>
        <apt:ExtendedFields>
                <act:DateData Name="_USERFIELD7">2004-01-10</act:DateData>
                <act:BooleanData Name="_USERFIELD1">false</act:BooleanData>
                <act:BooleanData Name="_USERFIELD4">false</act:BooleanData>
                <act:BooleanData Name="_USERFIELD5">true</act:BooleanData>
                <act:BooleanData Name="_USERFIELD6">true</act:BooleanData>
                <act:BooleanData Name="_USERFIELD9">false</act:BooleanData>
        </apt:ExtendedFields>
</apt:AddressData>
```

In a SOAP call, the Extended Fields are added directly at the end of a data structure in a dynamic List called "ExtendedFields". of "ObjectDataType". Because the WebService runs against a Schema (i.e. WSDL), the ExtendedFields contains a List of "ObjectDataType" elements for each Userfield. The Userfields are data-typed as they are defined in ABACUS.

The possible data-types for and ExtendedFields Element are :

- BooleanDataType – boolean type
- StringDataType – string type
- IntDataType – integer type

- LongDataType – long type
- DecimalDataType – decimal type
- DateDataType – data type

Note : The "DateTimeDataType" is not supported for ExtendedFields because a Userfield cannot be created with this data type in ABACUS.

Refer the the earlier section on "*Additional Parameters*" for information concerning the request of ExtendedFields via WebService. The referred section covers the usage of the "ACIncludeExtendedFields" parameter for the Find-Request operation.

Example : SOAP Save-Request from LOHN HierarchyEmployee Interface mit Extended Fields in 2 locations under the "EmployeeTimeAxis" and "AddressData" elements :

```
<?xml version="1.0" ?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Body>
  <ns4:SaveRequest xmlns:ns4="http://www.abacus.ch/abaconnect/2015.00/lohn/HierarchyEmployee"
xmlns:xmime="http://www.w3.org/2005/05/xmlmime"
xmlns:ns2="http://www.abacus.ch/abaconnect/2015.00/lohn/HierarchyEmployeeTypes"
xmlns="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
    <ns4:AbaConnectParam>
      <Login>
        <LoginToken>69a454826c88ad7d4919305df9be896c330a53e0</LoginToken>
      </Login>
      <Revision>0</Revision>
      <Level>Warning</Level>
    </ns4:AbaConnectParam>
    <ns4:Data>
      <ns2:Employee>
        <ns2:EmployeeNumber>63451</ns2:EmployeeNumber>
        <ns2:LastName>Mankins</ns2:LastName>
        <ns2:FirstName>Kevin</ns2:FirstName>
        <ns2:BadgeID>63451</ns2:BadgeID>
        <ns2:Contract>
          <ns2:Number>0</ns2:Number>
          <ns2:EmployeeTimeAxis>
            <ns2:YearTimeAxis>2015</ns2:YearTimeAxis>
            <ns2:MonthTimeAxis>1</ns2:MonthTimeAxis>
            <ns2:PersonalData>
            <ns2:DateOfBirth>1975-03-09</ns2:DateOfBirth>
            <ns2:Sex>M</ns2:Sex>
            <ns2:LanguageCode>D</ns2:LanguageCode>
            <ns2:CantonOfDomicile>SG</ns2:CantonOfDomicile>
            <ns2:Nationality>CH</ns2:Nationality>
            </ns2:PersonalData>
            <ns2:SettlementElements>
              <ns2:Country>CH</ns2:Country>
              <ns2:Currency>CHF</ns2:Currency>
            </ns2:SettlementElements>
            <ns2:ExtendedFields>
              <IntData Name="_USERFIELD7">1989</IntData>
              <StringData Name="_USERFIELD6">ZH 276 453</StringData>
            </ns2:ExtendedFields>
          </ns2:EmployeeTimeAxis>
          <ns2:EntryLeaving>
            <ns2:EntryDate>2015-11-27</ns2:EntryDate>
          </ns2:EntryLeaving>
          <ns2:PayrollData>
            <ns2:PayrollDataAndProperties>
              <ns2:Number>14</ns2:Number>
               <ns2:Value>10.0</ns2:Value>
               <ns2:DataType>VALUE</ns2:DataType>
               <ns2:ValidFrom>2015-01-01</ns2:ValidFrom>
             </ns2:PayrollDataAndProperties>
             <ns2:YearTimeAxis>2015</ns2:YearTimeAxis>
             <ns2:MonthTimeAxis>1</ns2:MonthTimeAxis>
          </ns2:PayrollData>
        </ns2:Contract>
        <ns2:AddressData>
          <ns2:AddressNumber>0</ns2:AddressNumber>
```

```
        <ns2:Country>CH</ns2:Country>
        <ns2:ZIP>9000</ns2:ZIP>
        <ns2:City>St. Gallen</ns2:City>
        <ns2:Language>D</ns2:Language>
        <ns2:Line1>Seestrasse 155</ns2:Line1>
        <ns2:TextField3>AC_27.11.2015 12.03:30</ns2:TextField3>
        <ns2:AddressValidAsOf>2015-11-27</ns2:AddressValidAsOf>
        <ns2:ExtendedFields>
          <DateData Name="_USERFIELD7">2004-01-10</DateData>
          <BooleanData Name="_USERFIELD1">false</BooleanData>
          <BooleanData Name="_USERFIELD4">false</BooleanData>
          <BooleanData Name="_USERFIELD5">true</BooleanData>
          <BooleanData Name="_USERFIELD9">false</BooleanData>
        </ns2:ExtendedFields>
      </ns2:AddressData>
    </ns2:Employee>
  </ns4:Data>
 </ns4:SaveRequest>
</S:Body>
</S:Envelope>
```

Please note that the name of a Userfield is iternally defined in ABACUS and linked to a database table. The name of the Userfield is unique within the table structure, but multiple tables may use the same internal name "_USERFIELDxx" to reference a Userfield. That means, via the AbaConnect Interfaces, that an identically named field with "_USERFIELDxx" may occur in two different "ExtendedFields" sub-structures, which references a different Userfield table.

Using Attachments in AbaConnect WebServices

An AbaConnect Interface may or may not support the transfer of Binary data files, generally called "Dossiers". If an AbaConnect interface supports dossiers, the files can be transferred as attachments when the main data is imported and exported via the interface.

The following prerequisites are required to transfer Dossiers via AbaConnect interfaces :

1. The interface must support Dossier transfers. This is shown in the AbaConnect Interface Summary available on the ABACUS Homepage.
2. The Abacus "Archivierung" must be installed and licenced
3. The Abacus User must be an "Archivierungs-User"
4. The Mandant must be activated for "Archivierung" (AbaTools-Mandant-Freigeben)

Dossiers are transferred as attachments via the AbaConnect interfaces. There are slight technical differences how actual attachment files are transferred via the AbaConnect Menu and via WebServices, but the concepts and reference to the Dossier documents within the XML structure are the same in both cases. Therefore it helps to look at concepts used via the ABACUS AbaConnect Menu in Program 625, also when using WebService.

The AbaConnect WebServices use the MTOM Standards to transfer binary data. These Standards define that the binary data is transfered as an external attachment, und not "inline" in the data stucture itself. (MTOM = Message Transmission Optimization Mechanism : https://www.w3.org/TR/soap12-mtom)

The binary files are transferred via AbaConnect WebService in an "Attachment" element

block. The "Name" field of the Attachment element block is linked to a main data structure via a reference field. The examples below show the main methods used in the AbaConnect interface to reference the Attachments to the main data structure.

An attachment is transferred as an encoded entity such as Base64, etc., and not using a link to a file path name on the local disk. The "Name" of the attachment is generally the short file name of the transferred file, without the path.

Kred-Document Interfaces with DocumentPicture Field

In the Kred-Document interface, a single binary attachment can be transferred. The "DocumentPicture" field links the Document to the Attachment via the "Name" field, and contains the short file name (without the folder path). The "DocumentPicture" field must correspond to the "Name" in the Attachment, so that the Binary data can be referenced to the main data structure.

```
<?xml version="1.0" ?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Body>
   <ns4:SaveRequest xmlns:ns4="http://www.abacus.ch/abaconnect/2013.00/kred/kreddocument"
xmlns:xmime="http://www.w3.org/2005/05/xmlmime" xmlns:ns2="http://www.abacus.ch/abaconnect/2013.00/kred/kreddocumentTypes"
xmlns="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
      <ns4:AbaConnectParam>
         <Login>
            <LoginToken>a027b697e3da71c9b85bc6fb97a2d7188747bd17</LoginToken>
         </Login>
         <Revision>0</Revision>
         <Level>Warning</Level>
      </ns4:AbaConnectParam>
      <ns4:Data>
         <ns2:Document>
            <ns2:AccountPayableDispositionDate>2015-01-08T00:00:00.000+01:00</ns2:AccountPayableDispositionDate>
            <ns2:AccountPayableDocumentDate>2015-01-08T00:00:00.000+01:00</ns2:AccountPayableDocumentDate>
            <ns2:Amount>1238.7</ns2:Amount>
            <ns2:CreditAccount>2000</ns2:CreditAccount>
            <ns2:Currency>CHF</ns2:Currency>
            <ns2:CustomDocumentDeadLine>30</ns2:CustomDocumentDeadLine>
            <ns2:DiscountDays1>0</ns2:DiscountDays1>
            <ns2:DiscountPercentage1>0.0</ns2:DiscountPercentage1>
            <ns2:DocumentCode>F</ns2:DocumentCode>
            <ns2:DocumentPicture>1449571013149.pdf</ns2:DocumentPicture>
            <ns2:GeneralLedgerDate>2015-01-08T00:00:00.000+01:00</ns2:GeneralLedgerDate>
            <ns2:KeyAmount>1238.7</ns2:KeyAmount>
            <ns2:LineItem>
               <ns2:Account>1200</ns2:Account>
               <ns2:Amount>1238.7</ns2:Amount>
               <ns2:CostCentre1>0</ns2:CostCentre1>
               <ns2:Number>1</ns2:Number>
               <ns2:Quantity>0.0</ns2:Quantity>
               <ns2:TaxCategory>2</ns2:TaxCategory>
               <ns2:TaxCode>111</ns2:TaxCode>
               <ns2:TaxRate>7.6</ns2:TaxRate>
               <ns2:Text>AC LineItem Text</ns2:Text>
            </ns2:LineItem>
            <ns2:ReferenceNumber></ns2:ReferenceNumber>
            <ns2:StatusIdentification>OFFEN</ns2:StatusIdentification>
            <ns2:SupplierNumber>18</ns2:SupplierNumber>
            <ns2:Text>AC_08.12.2015 11.36:53</ns2:Text>
         </ns2:Document>
         <ns2:Attachment>
            <ns2:BinaryData>
               <ns2:Name>1449571013149.pdf</ns2:Name>
               <ns2:BinData xmime:contentType="application/octet-stream">
                  <xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include" href="cid:a2b91634-7bc0-4a55-8355-
934cbe871205@example.jaxws.sun.com"></xop:Include>
               </ns2:BinData>
            </ns2:BinaryData>
         </ns2:Attachment>
      </ns4:Data>
   </ns4:SaveRequest>
```

```
</S:Body>
</S:Envelope>
--uuid:4fce506a-40df-4c68-a243-2dcc8cc1e921
Content-Id: <a2b91634-7bc0-4a55-8355-934cbe871205@example.jaxws.sun.com>
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

%PDF-1.4
%Ã¤Ã¼Ã¶ÃŸ
2 0 obj
.....
..... usw.
```

Address interfaces with Portrait Picture and Dossier

The Address interfaces from 2016.00, can contain both a PortraitPicture and one or more Dossier Elements. The PortraitPicture is a single element in the main data structure, used to define the link to a single attachment. The Dossier-element is used to reference the various Dossier associated with the Address record. Dossier-elements can occur more than once, with each Dossier-Element referencing one attachment.

For each "BinaryData" element in the Attachment element block, there should be a corresponding link to an element in the main data structure (e.g. "BinaryData" is referenced by a Dossier element, PortraitPicture element field, etc.). The link between the "BinaryData" element and the main data structure is alway via the "Name". If an Attachment is sent with a Save-Requests (Import) and not referenced via the Name in the main XML data structure then it **will not** be linked to the data-record.

```
<?xml version='1.0' encoding='utf-8'?><soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body>
   <FindResponse xmlns="http://www.abacus.ch/abaconnect/2016.00/adre/address"
xmlns:act="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes"
xmlns:apt="http://www.abacus.ch/abaconnect/2016.00/adre/addressTypes">
      <ResponseMessage>
         <act:IsFinished>true</act:IsFinished>
         <act:Bookmark>45C6D00F4F283AF2</act:Bookmark>
      </ResponseMessage>
      <DataContainer>
         <Data>
            <apt:Address>
               <apt:AddressNote>
                  <apt:NoteNumber>0</apt:NoteNumber>
                  <apt:Text>Some text to import for Jeremy Grace</apt:Text>
               </apt:AddressNote>
               <apt:AddressNumber>8140314764</apt:AddressNumber>
               <apt:LastName>Grace</apt:LastName>
               <apt:FirstName>Jeremy</apt:FirstName>
               <apt:CodeName>GRACE JEREMY</apt:CodeName>
               <apt:Country>CH</apt:Country>
               <apt:ZIP>9001</apt:ZIP>
               <apt:City>St. Gallen</apt:City>
               <apt:Language>D</apt:Language>
                .....
               <apt:PortraitPicture>ADR8140314764.bmp</apt:PortraitPicture>
               <apt:ConditionProposal>0</apt:ConditionProposal>
               <apt:AddressDossier>
                  <apt:Name>KORRESPONDENZ</apt:Name>
                  <apt:Type>dossier</apt:Type>
                  <apt:ArchiveYear>0</apt:ArchiveYear>
                  <apt:Description>AC-Document 2017-08-18 14:03:14</apt:Description>
                  <apt:Application>adre</apt:Application>
                  <apt:Object>75</apt:Object>
                  <apt:UDK>8140314764</apt:UDK>
                  .....
                  <apt:FileSize>96854</apt:FileSize>
```

```
                  <apt:FileName>5_75_korrespondenz_8140314764_tmp_83559.jpg</apt:FileName>
                  <apt:OriginalFilename>tmp_83559.jpg</apt:OriginalFilename>
               </apt:AddressDossier>
            </apt:Address>
            <apt:Attachment>
               <apt:BinaryData>
                  <apt:Name>ADR8140314764.bmp</apt:Name>
                  <apt:BinData>
                     <xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include" href="cid:ADR8140314764.bmp"></xop:Include>
                  </apt:BinData>
               </apt:BinaryData>
               <apt:BinaryData>
                  <apt:Name>5_75_korrespondenz_8140314764_tmp_83559.jpg</apt:Name>
                  <apt:BinData>
                     <xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include"
href="cid:5_75_korrespondenz_8140314764_tmp_83559.jpg"></xop:Include>
                  </apt:BinData>
               </apt:BinaryData>
            </apt:Attachment>
         </Data>
      </DataContainer>
   </FindResponse>
</soapenv:Body>
</soapenv:Envelope>
--MIMEBoundary_26425012118a37e511dd16b932ebe8c9c85726d82cebb6df
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
Content-ID: <ADR8140314764.bmp>

....(Attention : binary data truncated for display !!)
```

Central-Dossier interfaces and interfaces which implement the Dossier element

The Central Dossier interface allows Dossiers from the Abacus Application and also central Dossiers to be exported and imported. The Dossier must already exist in Abacus – Dossiers cannot be created via the AbaConnect interfaces. Some of the AbaConnect Application interfaces implement a sub-set of the Dossier-Element in the data structure, which allows Dossiers related to a particular data record to be transferred (e.g. "HierarchyEmployee_201x_00", from "address_2016_00" and "Activities_2016_00", plus various other interfaces). Please see the documentation "AbaConnect Schnittstellen Zusammenfassung" on the Abacus Homepage to see if an interface support Dossiers

When using the Central Dossier interface to access Application dossiers for a particular data record, the UDK field muss be correctly defined. The UDK field is an Application defined link to the data record. The best way to examine the UDK details is to create an export XML file via AbaConnect in the Abacus Menu, to see how the data records are link to the dossiers.

The AbaConnect WebService examples available on the Abacus Homepage, also show how Dossiers can be transferred via the AbaConnect interfaces. Note that the data export and WebService Find-Requests also provide good examples of how the Dossier data is formatted, for the import and WebService Save-Requests.

Multiple Data Records with Find Request

As of Abacus V2021 it is possible to export multiple data records with a single Find Request when navigating using the Operations FIRST, NEXT, LAST and PRIOR using the additional parameter "ACDataRecordCount". The parameter is an integer value between

1 and 50. A maximum of 50 records can be exported with a single Find Request. The default value is 1 (i.e. one data record per request). For Find Request using GREATER_EQUAL and EQUAL, the returned data count is limited to a single record. By further navigation with Find.NEXT the "ACDataRecordCount" can be used.

The procedure of navigating through records with Find.NEXT and "Bookmark" is identical to the existing implementations with single records. The difference is the number of data records that are returned in the Response.

The existing WSDL's for all previous WebServices already support the return of multiple data records. Updated WSDL's on the Abacus Homepage (from interface version 2018.00) may also support the additional Attributes, "bookmark" and "dataIndex", on the Data-Element (see data structure below). The additional attributes can be used to help iterate through the Data Elements, if required.

Depending on the AbaConnect interface the export of multiple data records can improve the performance. Generally, a recommended Data Count is 10 - 20 records per request. The performance difference between 20 and 50 data records per request is not significant.

Note : Export of Dossiers (e.g. Binary attachments such as PDF's and images) is only supported with single data records. Dossiers cannot be exported when using mutlitple data records. Request for Dossiers will be automatically deactivated when retrieving multiple data records. If the request explicitly activates the Dossier (e.g. using "ACGAPDossiersInclude" = true), an error message will be returned.

AbaConnect Dossier interfaces (such as “centraldossier_20xx_00”, etc.) that only export Dossiers do not support the use of "ACDataRecordCount"

Example Find Request with multiple data record count

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <ns1:FindRequest xmlns:ns1="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes">
         <ns1:AbaConnectParam>
            <ns1:Login>
               <ns1:LoginToken>ef2eb940c11cea18ddf14cadf971e70667c7102edcdb29be2d8ca64c88b73264</ns1:LoginToken>
            </ns1:Login>
            <ns1:Revision>0</ns1:Revision>
            <ns1:Level>Warning</ns1:Level>
            <ns1:Additional>
               <ns1:IntData Name="ACDataRecordCount">10</ns1:IntData>
            </ns1:Additional>
         </ns1:AbaConnectParam>
         <ns1:FindParam>
            <ns1:Index>1</ns1:Index>
            <ns1:Operation>FIRST</ns1:Operation>
         </ns1:FindParam>
      </ns1:FindRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example Find Response with multiple data records

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header></soapenv:Header>
   <soapenv:Body xmlns="http://www.abacus.ch/abaconnect/2020.00/adre/address"
xmlns:act="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes"
xmlns:apt="http://www.abacus.ch/abaconnect/2020.00/adre/addressTypes">
      <IsFinishedResponse>
         <ResponseMessage>
```

```
    <act:IsFinished>true</act:IsFinished>
    <act:Bookmark>00000000000000003CF07B214D283AF2</act:Bookmark>
    <act:Messages></act:Messages>
  </ResponseMessage>
  <DataContainer>
    <Data bookmark="00000000000000033F07B214D283AF2" dataIndex="0">
      <apt:Address>
        <apt:AddressNumber>1</apt:AddressNumber>
        <apt:LastName>Abacus Demo AG</apt:LastName>
        <apt:CodeName>ABACUS DEMO AG</apt:CodeName>
          ....
          ....
      </apt:Address>
    </Data>
    <Data bookmark="00000000000000034F07B214D283AF2" dataIndex="1">
      <apt:Address>
        <apt:AddressNumber>2</apt:AddressNumber>
        <apt:LastName>Matter &amp;amp; Schlatter</apt:LastName>
        <apt:CodeName>MATTER &amp;amp; SCHLATT</apt:CodeName>
        <apt:Country>CH</apt:Country>
          ....
          ....
      </apt:Address>
    </Data>
    <Data bookmark="00000000000000035F07B214D283AF2" dataIndex="2">
      <apt:Address>
        <apt:AddressNumber>3</apt:AddressNumber>
        <apt:LastName>Rilling AG</apt:LastName>
        <apt:CodeName>RILLING AG</apt:CodeName>
          ....
          ....
      </apt:Address>
    </Data>
    <Data bookmark="00000000000000036F07B214D283AF2" dataIndex="3">
      <apt:Address>
        <apt:AddressNumber>4</apt:AddressNumber>
        <apt:LastName>Swisscom AG</apt:LastName>
        <apt:CodeName>SWISSCOM AG</apt:CodeName>
          ....
          ....
      </apt:Address>
    </Data>
    <Data bookmark="00000000000000037F07B214D283AF2" dataIndex="4">
      <apt:Address>
        <apt:AddressNumber>5</apt:AddressNumber>
        <apt:LastName>Lagro GmbH</apt:LastName>
        <apt:CodeName>LAGRO GMBH</apt:CodeName>
          ....
          ....
      </apt:Address>
    </Data>
    <Data bookmark="00000000000000038F07B214D283AF2" dataIndex="5">
      <apt:Address>
        <apt:AddressNumber>6</apt:AddressNumber>
        <apt:LastName>Rockwell Inc.</apt:LastName>
        <apt:CodeName>ROCKWELL INC</apt:CodeName>
          ....
          ....
      </apt:Address>
    </Data>
    <Data bookmark="00000000000000039F07B214D283AF2" dataIndex="6">
      <apt:Address>
        <apt:AddressNumber>7</apt:AddressNumber>
        <apt:LastName>Dolbi Ltd</apt:LastName>
        <apt:CodeName>DOLBI LTD</apt:CodeName>
          ....
          ....
      </apt:Address>
    </Data>
    <Data bookmark="0000000000000003AF07B214D283AF2" dataIndex="7">
      <apt:Address>
        <apt:AddressNumber>8</apt:AddressNumber>
        <apt:LastName>Bosshard AG</apt:LastName>
        <apt:CodeName>BOSSHARD AG</apt:CodeName>
        <apt:Country>CH</apt:Country>
          ....
```

```
                ....
            </apt:Address>
          </Data>
          <Data bookmark="00000000000000003BF07B214D283AF2" dataIndex="8">
            <apt:Address>
              <apt:AddressNumber>9</apt:AddressNumber>
              <apt:LastName>Strasser</apt:LastName>
              <apt:FirstName>Karl</apt:FirstName>
              <apt:CodeName>STRASSER KARL</apt:CodeName>
                ....
                ....
            </apt:Address>
          </Data>
          <Data bookmark="00000000000000003CF07B214D283AF2" dataIndex="9">
            <apt:Address>
              <apt:AddressNumber>10</apt:AddressNumber>
              <apt:LastName>Krupp Edelstahl</apt:LastName>
              <apt:CodeName>KRUPP</apt:CodeName>
                ....
                ....
            </apt:Address>
          </Data>
        </DataContainer>
      </IsFinishedResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

The central "ResponseMessage" – "Bookmark" can be used to navigate through the data records, the same as navigating through the data records with single records. The "ResponseMessage" – "Bookmark" references the last data record returned. When navigating the data records with Find LAST or PRIOR, the data records will be in the reverse order.

The number of data records can be less than the requested data count, when no more data is available. Navigating past the last data set will return the existing "8050" Warning message, as shown below :

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header></soapenv:Header>
   <soapenv:Body xmlns="http://www.abacus.ch/abaconnect/2018.00/adre/address"
xmlns:act="http://www.abacus.ch/abaconnect/2007.10/core/AbaConnectTypes"
xmlns:apt="http://www.abacus.ch/abaconnect/2018.00/adre/addressTypes">
      <FindResponse>
        <ResponseMessage>
          <act:IsFinished>true</act:IsFinished>
          <act:Messages>
            <act:Message>
              <act:Code>8050</act:Code>
              <act:Text>Es wurden keine Daten gefunden. Bitte überprüfen Sie die Suchargumente.</act:Text>
              <act:Level>Warning</act:Level>
            </act:Message>
          </act:Messages>
        </ResponseMessage>
      </FindResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Using Secure SSL Connections with AbaConnect WebServices

A connection with SSL via WebService is a configuration that does not really have anything to do with the funtionality of the WebService. The WebService will work the same in insecure connection as with a secure connection.

If the SSL configuration is incorrect, then the WebService will not be able to connect successfully via WebService. This can be due to an incorrect Server configuration or an incorrect Client configuration.

Client Side SSL Configurations

The first simple checks to do with WebService connections are similar to insecure connections via the URL in Internet Browser normally via Port 443:

- https://<abacus-server-domain>:443
  This should show the Abacus Login Menu Page

- https://<abacus-server-domain>:443/abaconnect/services
  This should show the AbaConnect Welcome message :
  e.g. *"Welcome to AbaConnect-Webservices. For more Information see http://www.abacus.ch"*

Note : If either on of the above tests fails, then a connection via AbaConnect WebServices is also not possible.

Configuration of Client-Side SSL Certificates

The best place to find information on SSL Connection configuration is the AbaConnect WebService examples. The source code contains details how the SSL configuration is configured in .NET and JAVA. The two development environment differ in how the SSL Certificate is configured. The AbaConnect WebService examples have an option check box to "Use SSL" in the example program user interface (next to the Server Name and Port input fields). Generally the available default configurations will be sufficient to configure an SSL connection with official SSL Certificate, but is depends on the certicate source. Official SSL Certificates obtained from registered organisations, should generally already be recognized by the various development environments (e.g. .NET or JAVA). If the SSL Certificate is not recognized (ie. Self-signed certificates), then the certificate has to be entered as a "trusted" certifcate. How this is done depends on the development environment.

Programs running with .NET Framework
Programs running with the .NET Framework, normally use the Windows based components to access the "trusted" store. Often the certificate can be accepted as "trusted" in other Windows programs such as "Internet Explorer" or "Microsoft Edge". The "trusted" certificates should also be referenced by the .NET Framework. Other possibilities may include registering the SSL Certificate directly in the available .NET components within the Code (Please refer to Microsoft documentation sources). Normally, the option via the Internet Explorer or Edge is the simplest option.

Programs running with JAVA:
Every JAVA installlation has its own "TrustStore". The default JAVA "Truststore" installed with JAVA should include most official SSL certificates, sourced from official organisations. Other SSL certifcates must be manually imported into the JAVA "TrustStore". This can be done with the JAVA "keytool.exe" program (e.g. *"c:\Program Files\Java\jre1.8.0_191\bin\keytool.exe"*).

The SSL Certicate must be imported into the JAVA "TrustStore" of the same JAVA version

used to run the Cleint Program. The default JAVA "TrustStore" is generally available in the JAVA security folder (e.g. *"c:\Program Files\Java\jre1.8.0_191\lib\security\cacerts"*).

Common practice is to import the additional SSL Certificates into a separate "private" certificate file (e.g. *"C:\Program Files\Java\jre1.8.0_191\lib\security\private_certs"*), which is then configured in the JAVA Client Program.

Interpreting Exceptions due to SSL Problems
If the SSL Certificate is invalid or incorrectly configured, the resulting Exception that occurs can be sometime difficult to interpret. It can sometimes be helpful to search the internet with the error/exception message, to find similar problems and solutions.

Also using the AbaConnect WebService Example programs or the AbaConnect Ping-Tester program, can offer an independent check, and help to check the SSL configurations. The simplest AbaConnect method to test the connection is the Ping-Request. If the Ping-Request is successful, then other AbaConnect Methods such as Login, Login, etc. should also work successfully.