SIEMENS

SIMATIC IT Unilab 6.7

The XML Interface and Web Service

Concepts and User Manual

Edition 09/2014

A5E00436150-09

Guidelines

This manual contains notices intended to protect the products and connected equipment against damage. These notices are graded according to severity by the following texts:

Caution

Indicates that if the proper precautions are not taken, this can result into property damage.

Notice

Draws your attention to particularly important information on handling the product, the product itself or to a particular part of the documentation.

Trademarks

Disclaimer of Liability

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E00436150-09
09/2014

Preface

Where is this manual valid?

This manual is valid for release 6.7 of SIMATIC IT Unilab.

Basic knowledge required

This guide is intended for SIMATIC IT Unilab users who are responsible for system configuration, such as application managers and system integrators (consultants). To be able to understand the concepts and examples discussed in this guide, the reader should at least have taken the SIMATIC IT Unilab Basic Training.

Purpose

This Concepts and User Manual explains how the XML interface and the XML web service in Unilab work.

Related documentation

The Unilab Concepts Guides (part 1, 2 and 3) contain information related to the content of this Concepts and User Manual.

These documents are all available online from the SIMATIC IT Unilab Documentation Library**.**

Conventions

The table below describes the specific typographic conventions that are used throughout this manual:

Symbol/Convention	**Indicates...**
E.g.	Where examples are given.
Text in bold	The names of menus, commands, dialog boxes and toolbar buttons and, in general, all strings (e.g. **File** menu; **Save** command).
KEY1+KEY2	Shortcut keys, which permit rapid access to commands (e.g. CTRL+C).
UPPERCASE	The names of keyboard keys (e.g. RETURN key).
Italics	Noun with special importance or significance for which emphasis is needed. The names of parameters that must be replaced with a specific name or value.
>	A succession of commands in which the command preceding the symbol must be selected before the

Symbol/Convention	Indicates...
	command following it.
`Code example`	Code example.

SIMATIC IT Documentation Library

The SIMATIC IT Documentation Library provides you with a comprehensive and user-friendly interface to access the overall product documentation where manuals and helps online can be browsed by functionality or by component.

Readme

The installation includes a readme file, which contains information on upgrade procedures and compatibility with previous releases. This file is supplied both in standard text (**Readme.wri**) and in Acrobat PDF (**Readme.pdf**) format.

This file is available in folder \ReleaseNotes of the setup DVD and is available from the SIMATIC IT Unilab Documentation Library.

Acronyms and abbreviations

The table below lists the acronyms and abbreviations that are used throughout this manual:

Acronyms / Abbreviation	Meaning
API	Application Program Interface
au	Attribute
cf	Custom function
db	Database
DBA	Database Administrator
DLL	Dynamic Link Library
ev	Event
gk	Group key
ic	Info card
id	Identification
ie	Info field (Configuration)
ii	Info field (Operational)
ip	Info profile
lc	Life cycle (or life cycle model)
LIMS	Laboratory Information Management System
me	Method (Operational)
MES	Manufacturing Execution System
mt	Method (Configuration)
pa	Parameter (Operational)
pg	Parameter group
pp	Parameter profile

Acronyms / Abbreviation	Meaning
pr	Parameter Definition
pref	Preference
rq	Request
rt	Request type
sc	Sample code
seq	Sequence (number)
st	Sample type
up	User profile
us	User

SIMATIC IT Training Center

Siemens IA AS MES offers a number of training courses to familiarize you with the SIMATIC IT product suite. To successfully achieve this goal, training consists of lessons in both theory and practice.

Courses are held year-round, according to a program that is published well in advance of the first scheduled session.

The material on the basis of which our courses are conducted reflects the result of years of experience in process, LIMS, quality control and production management.

All courses are held by expert personnel that are aware of the developments and innovations in the Siemens IA AS MES product suite.

Courses are held in English at the Siemens IA AS MES Training Centers.

Upon request, training courses can also be organized on the customer's premises.

For more information on the training course calendar, please visit our technical web site (http://www.siemens.com/simatic-it/training).

SIMATIC IT Service & Support

A comprehensive Software Maintenance program is available with SIMATIC IT products. Software Maintenance includes the following services:

- **Software Update Service** (SUS): automatic distribution of upgrades and service packs
- **Technical Support Service** (TSS): support on technical problems with SIMATIC IT software (standard support and other optional services)
- **Online Support**: a technical web site, providing information such as Frequently Asked Questions and technical documentation on SIMATIC IT products

Software Update Service (SUS)

This service provides automatic shipment of new versions and service packs when released. When a new version / service pack is available for shipping, it is typically shipped within one month.

One copy of the installation DVD is shipped for each Server covered by Software Maintenance.

Hot fixes (officially tested and released) are not shipped and must be downloaded from the Technical Support Service web site.

Technical Support Service (TSS)

Siemens provides a dedicated technical support team for SIMATIC IT products.

The following options are available:

Bronze support: 9 hours/day, 5 days/week

Silver support: 24 hours/day, 5 days/week

Gold support: 24 hours/day, 7 days/week

The principal language of the SIMATIC IT hotline is English.

SIMATIC IT partners and customers covered by the Software Maintenance program are entitled to direct access to the TSS.

Access to TSS

To be able to access TSS, the customer needs to register as a user on the Technical Support web site. Connect to http://www.siemens.com/mes-simaticit/ and follow the **Technical Support Service** link.

The registration form must be completed with:

- Personal data
- The required company and plant information
- The Contract Number provided by Siemens Back Office when the contract is agreed.

Online Support

A customer who is a registered TSS user, can access the Technical Support web site (http://www.siemens.com/mes-simaticit/tss), which contains technical information such as:

- Service conditions (Phone numbers, Working hours, Reaction times,…)
- SIMATIC IT knowledge base: a technical support database that includes practical service solutions from Technical Support or the SIMATIC IT community
- SIMATIC IT software (e.g. hot fixes, software examples) and release notes that can be downloaded
- SIMATIC IT cross-industry libraries that can be downloaded (limited access to SIMATIC IT certified partners)
- SIMATIC IT product documentation that can be downloaded
- Frequently Asked Questions and useful tips.

Table of Contents

1 Description and Architecture

1.1 About this Manual

This manual provides you with all necessary information about the *SIMATIC IT Unilab XML Interface*. This manual refers to this application as the *XML Interface.*

It covers the following topics:

Topic	See Chapter
Description and Architecture	1
The XML Interface	2
Import/Export Data	3
The XML Web Service	4
Exchanging Data between Servers	5

1.2 Description and Architecture

The **SIMATIC IT Unilab XML Interface** allows importing and exporting SIMATIC IT Unilab data in XML format.

This import/export can be triggered in two ways:

- Interactively with the client applications
- By putting xml-files with import/export requests in a dedicated folder.

The system uses four dedicated data folders:

- IN: contains all XML files to be processed
- OUT: contains all processed XML files
- LOG: contains copies of processed files
- ERROR: contains copies of files for which errors occurred during processing.

1.3 Related Functions and Applications

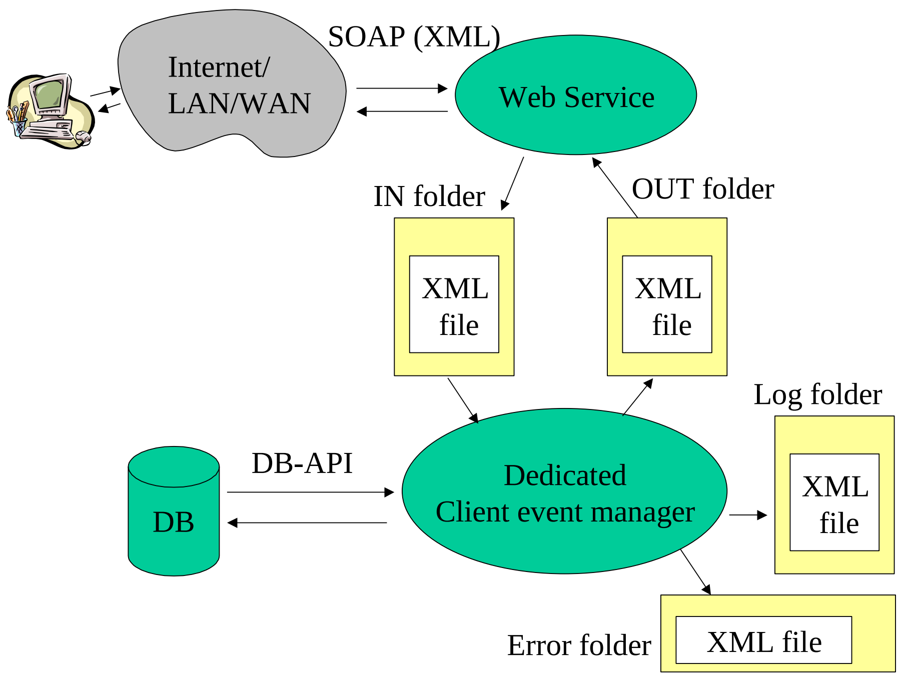
Several SIMATIC IT Unilab functions are based on the SIMATIC IT XML Interface:

Function	Description
SIMATIC IT Unilab XML Web Service	This web service enables the import/export of SIMATIC IT Unilab data in XML format via a web service.
SIMATIC IT Unilab Interface for SAP QM	This interface allows the exchange of quality data between SIMATIC IT Unilab and SAP. A full description of this interface is beyond the scope of this document. For more information, refer to the *SIMATIC IT Unilab Interface for SAP QM* manual.
Direct Exchange	The direct exchange of SIMATIC IT Unilab data between different servers, via menu bar entries in the Analyzer and Configuration application.
Data Import	Import of the demo data sets EasyQC and EasySL.

1.4 Architecture

1.4.1 Diagram

The diagram below shows the architecture of *SIMATIC IT Unilab XML Interface and Web Service*.



1.4.2 Process Description

The Process

The SIMATIC IT Unilab XML Interface and Web Service process is described below:

Phase	Description
1	The *Client Event Manager (CEM)* checks a dedicated folder ("*IN*") for new XML files. These files contain requests for: • Exporting data from the Unilab database (export request), or • Importing data into the Unilab database (import request). **Note:** All files are handled sequentially, in alphabetical order.
2	In case of an export request, the CEM • Retrieves the required information from the database • Encloses it in an XML file • Saves the XML file in the OUT folder. This file has the same name as the input file. If so configured, the system applies an additional XSLT transformation on the retrieved data.
3	In case of an import request, the XML Interface tries to import the data into the database.

1.4.3 Data import

Description

If the XML file contains data that needs to be imported, the XML Interface will try to save the data in the database.

All data in a single file is handled in a single transaction. This means that the import is done either for all objects in the file, or for none of them.

Note

When there is no need to consider the data import as a single transaction, it is a good practice to make one file for each imported object.

What happens after processing depends on whether or not the file was successfully processed.

For more information, refer to *"Data Import"*.

1.5 Web service

The XML Interface is also available through a **.NET** web service.

This web service:

- Handles incoming SOAP requests.
- Uses the https protocol (with SSL – Secure Socket Layer).

- Checks the authentication of the user
- Validates the XML data with the SIMATIC IT Unilab XML scheme.
- Passes the request to the XML Interface.
- For a data export request, returns the result as the **OUT** parameter of a synchronous call.
- For a data import request, returns no feedback.

Note

Using the web service requires the installation of the Microsoft **.NET** framework. For more information, refer to SIMATIC IT Unilab – Prerequisites in the Unilab Product Library.

1.6 About security

The XML interface does not offer access control. Anyone with access to the interface folders can import or export data from and to the SIMATIC IT Unilab system using any valid Unilab user ID.

Therefore, we recommend restricting access to these folders: their direct use is meant only for interface applications such as the **SIMATIC IT Unilab SAP QM-IDI Interface**.

However, if you are using the **XML Web Service**, you can apply full security. This security is not based on the SIMATIC IT Unilab user/password combination. Instead, it is based on the use of signatures and certificates.

For more information, refer to “*The XML Web Service*”.

2 The XML Interface

2.1 Overview

This chapter covers the following topics:

Topic	See Section
Installation	2.2
The Unilab XML Schema	2.3
Configuration: Properties and Options	2.4

2.2 Installation

In order to enable the import\export of data as XML files using the CEM, select the option *Client Event Manager* during the client installation of SIMATIC IT Unilab.

2.2.1 Registry settings

The **ClientEvtMgr** section contains some dedicated registry settings for the XML interface. For details regarding these registry settings, please refer to the SIMATIC IT Unilab – Registry Settings manual in the Unilab Product Library.

2.3 The Unilab XML Schema

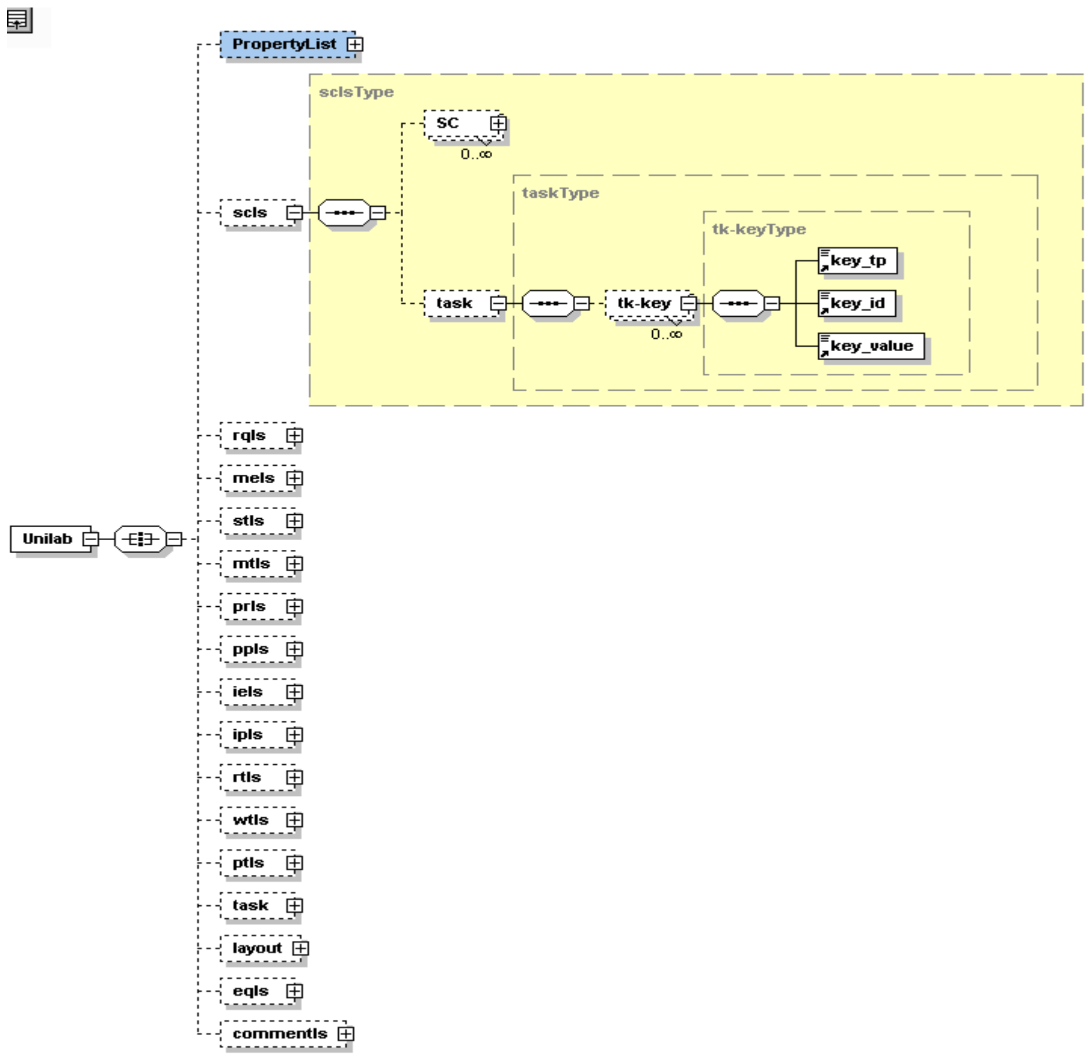
2.3.1 About the Unilab XML Schema

The Unilab XML schema describes the XML syntax in detail. This file contains all details regarding the supported schema, including enumerations of possible values and description of data types.

This complete schema is found in the *<Unilab Installation folder>\Doc* folder, in the *XmlWrapper.xsd* file.

2.3.2 Graphical View

We recommend using an XML tool to graphically view the xsd file. This simplifies its interpretation, as shown in the illustration below:



2.3.3 Some properties of the schema

Some Properties

This schema has the following properties:

- The schema defines the *xsdUnilab* name space.
- The root node is *xsdUnilab:Unilab*. Basically, this root node contains:
  - A *PropertyList* element with metadata and directives for the XML Interface. **Note:** The possible properties are enumerated in detail in section *2.4.3* Import/Export File Properties*.*
  - Nodes with lists of SIMATIC IT Unilab data.
- The requested action is specified by a *<prop>* node with attribute *name*=”*function*”. The list of supported functions can be found in the next sections.

Use of Uppercase and Lowercase

There is a difference between nodes with a node name in uppercase and those in lowercase. This difference is necessary for a correct XML schema.

Example: The **<PA>** node stands for the parameter node and the **<pa>** node for the parameter ID node.

2.4 Configuration: Properties and Options

The directives that control the behavior of the XML Interface can be inserted at three different levels:

Level	Description
In the PropertyList	As a *Property node.* Options set here apply to the entire XML-file
In the *<Object> ListNode (ls)*	As an attribute, options set here apply to the entire list.
In the *Object node* itself	As an attribute, options set here apply to only the node itself.

Examples

PropertyList Level

```
<prop name="ObjectLifeCycle">BG</prop>.
```

<Object> ListNode

```
<stls Objectstatus='@E'>
```

Object node

```
<ST IgnoreStatus ='Yes'>
```

2.4.1 Insertion Level Priorities

If an option is specified at different levels, the lowest level overrules. This means that the lowest level has the highest priority.

Therefore, from the highest level to the lowest, the priorities can be classified as:

Node	Level	Priority
Property	High	Low
List	Middle	Middle
Object	Low	High

2.4.2 Import/Export Options

The following table provides an overview of all options:

Option	**Description**	**Operational**	**Configuration**	**Import**	**Export**	**PropertyList property**	**Object Attribute**
handle_events	**Possible Values**: • Yes (default) • No If set to *No*, all events that would be initiated by the data import are dropped.	x	x	x	x	x	x
ObjectLifeCycle	Specifies the lifecycle of the imported objects	x	x	x	x	x	x
ObjectStatus	Specifies the status of the imported objects. A status change to the specified status is performed. **Possible Values**: Any valid status (e.g. @A, @C,…)If set to *@C*, the object will be cancelled as in the Client application. **Note:** This option does not have effect when *IgnoreStatus* is set to *Yes*.	x	x	x	x	x	x
IgnoreStatus	**Possible Values**: • Yes (default) • No If *Yes,* the status fields in the xml file are ignored. No status change is performed during the import.	x	x	x	x	x	x
OverwriteStatus (1)	**Possible Values**: • Yes • No (default) If set to *Yes,* and when the life cycle allows a status transition from, e.g. 'Approved' to 'In Editing', the object is imported without creating a new version of the object. If set to No, the object is ignored and an error message is displayed.	x	x	x	x	x	
AllObjMajorVersion (2)	If Yes, then new major versions will be created for all the imported configuration objects.	x	x	x	x	x	x

Table continues on next page…

Option	Description	Operational	Configuration	Import	Export	PropertyList property	Object Attribute
merge	**Possible Values**: • **only_new**: only new records will be added to the table. The primary key is used to determine which records are new. • **merge**: existing records with the same primary key will be updated; new records will be added. • **Replace**: rows in the existing table will be replaced with the new records. All existing records will be deleted. Note the *where_clause* node allows selecting the records that will be deleted. **Note**: This option only applies to *table/tablels* nodes.		x	x	x	x	
rqCreateRq (*)	**Possible Values**: • **Yes**: the *CreateRequest* DB API is executed • **No**: the *SaveRequest* DB API is executed • **Automatic**: if a definition of the sample type exists, the *SaveRequest* DB API is executed. If the sample definition does not exist, the *CreateRequest* DB API is executed **Note:** This option only applies for rq/rqls nodes		x	x		x	

Table continues on next page…

Option	Description	Operational	Configuration	Import	Export	PropertyList property	Object Attribute
scCreateSc (*)	**Possible Values**: • **Yes**: the *CreateRequest* DB API is executed • **No**: the *SaveRequest* DB API is executed • **Automatic**: If a definition of the request type exists, the *SaveRequest* DB API is executed. If the definition does not exist, the *CreateRequest* DB API is executed **Note**: This option only applies to sc/scls nodes	x		x		x	x
sdCreateSd (*)	**Possible Values**: • **Yes**: the *CreateStudy* DB API is executed • **No**: the *SaveStudy* DB API is executed • **Automatic**: If a definition of the study type exists, the *SaveStudy* DB API is executed. If the definition does not exist, the *CreateStudy* DB API is executed **Note**: This option only applies to sd/sdls nodes	x		x		x	x
scCreatePg	**Possible Values**: • Yes (default) • No Sets the *a_create_pg* parameter of the *CreateSample* API and indicates whether the parameter groups should be created during sample creation or not. If not set, the scCreatePg user preference is used. **Note**: This option only applies to sc/scls nodes	x		x		x	x

Table continues on next page…

Option	**Description**	**Operational**	**Configuration**	**Import**	**Export**	**PropertyList property**	**Object Attribute**
scCreateIc	**Possible Values**: • Yes (default) • No Sets the *a_create_ic* parameter of the *CreateSample* API and indicates whether the info cards should be created during sample creation or not. If not set, the scCreateIc user preference is used.	x		x		x	x
rqCreateIc	Sets the *a_create_ic* parameter of the *CreateRequest* API and indicates whether the info cards should be created during sample creation or not. If not set, the rqCreateIc user preference is used.	x		x		x	x
sdCreateIc	Sets the *a_create_ic* parameter of the *CreateStudy* API and indicates whether the info cards should be created during study creation or not. If not set, the sdCreateIc user preference is used.	x		x		x	x
sdCreateSc	Sets the *a_create_sc* parameter of the *CreateStudy* API and indicates whether the samples should be created during study creation or not. If not set, the sdCreateSc user preference is used.	x		x		x	x
modify_flag	• MOD_FLAG_INSERT • MOD_FLAG_CREATE This setting can only be set as attribute on <PG>, <PA> and <ME> nodes. This sets the modify flag of new assigned parameter groups, parameters and methods.	x		x			x
modify_reason	Allows the user to specify a reason for changing the status or life cycle of an object.	x	x	x			x
occurrence	**Possible Values**: Any integer value placed between square brackets. E.g. <PA occurrence="[2]"> Allows the user to specify a specific occurrence whn there are a number of objects that occur in duplo (or more) instances.	x		x			x

Note for the options marked with (*):

The names (and values) of this option have corresponding preferences. If the option is not specified, the actual preference values (corresponding to the default task of the current user) are applied.

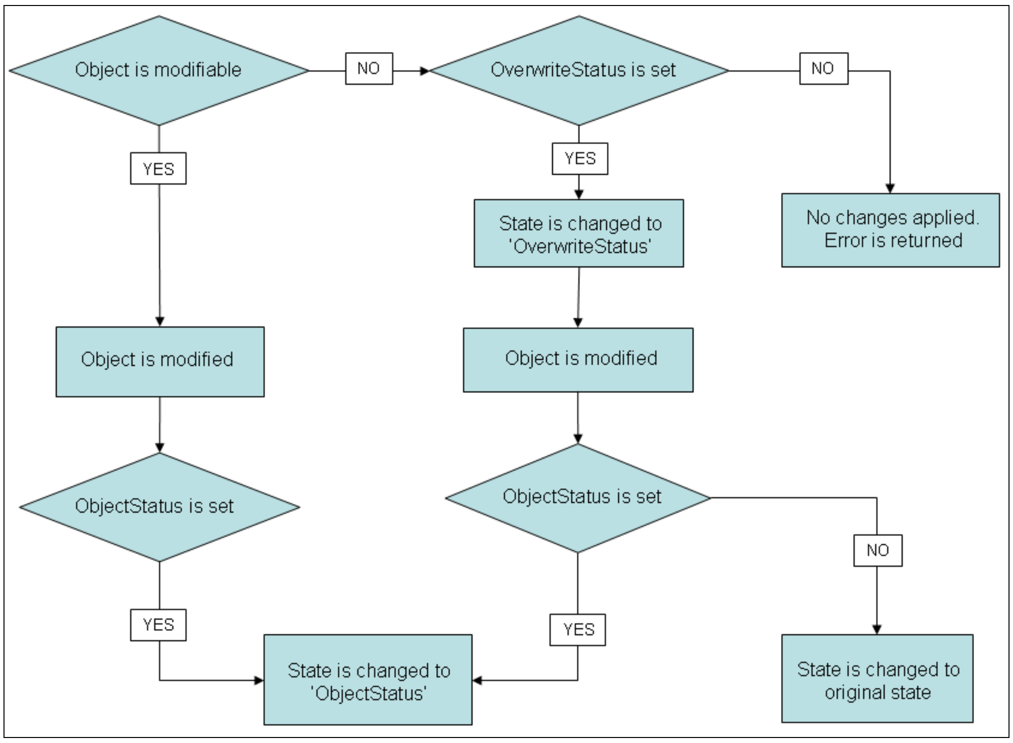
(1) OverwriteStatus Option

The XML interface must provide a way to overwrite existing, approved configuration objects, without creating a new version of the object.

Some life cycles can use the property OverwriteStatus to overwrite approved objects, and allowing a status transition from, e.g. '**Approved**' to '**In Editing**'.

When the life cycle allows this transition on a 21CFR 11 database, it is possible to apply changes to an approved version object via the XML interface.

When this property is set, the XML interface applies according to the following flow chart:


Object is modifiable
NO
OverwriteStatus is set
NO
YES
YES
State is changed to 'OverwriteStatus'
No changes applied. Error is returned
Object is modified
Object is modified
ObjectStatus is set
ObjectStatus is set
NO
YES
YES
State is changed to 'ObjectStatus'
State is changed to original state

(2) AllObjMajorVersion Option

For versions prior to Unilab 6.2, this option is referred as AllObjectMajorVersion. XML files where the option AllObjectMajorVersion is defined are processed correctly, and no error is generated.

2.4.3 Import/Export File Properties

The **PropertyList** element contains some metadata and directives for the XML Interface.

Property	Description
sid	Database identification
user_id	Unilab user id. Prior to the actual export or import, the system switches the database user to this user.
export_date	Date of the export
default_date_format	Default date format used during the export. Prior to the actual export or import, the system switches the default date format to this setting.
decimal_symbol	Decimal symbol used during the export
full_details	**Possible values**: *Yes* | *No* When set to *'Yes'*, then the child objects of the imported/exported objects will also be imported/exported. **Example**: When set to Yes, this property causes the XML Interface to import the *pp*s related to the *stpp* of an exported *st*.
time_zone	Support for multiple time zone, refers to timestamp with time zone. **Note**: For data exported with versions prior to Unilab 6.2, where this property is not set, it is used the time zone of the importing session.
Blocksize(*)	Specifies the maximum number of main objects an export file can contain. **Tip**: Use this setting to limit the size of the exported xml files. Large xml files may decrease performance seriously. **Example**: If the block size is set to 100 and the *selectSample* function returns 240 samples, then there will be three output files: two with 100 samples and one with 40 samples.

output_property(*)	This will set a prop node in the PropertyList of the output file.
	
	Example: Adding
	
	`<prop name="output_property">`
	`UseConfiguration=No</prop>`
	
	to the input file, will result in the property node
	
	`<prop name="UseConfiguration">No</prop>`
	
	in the output file.

function(*)	Specifies the function that will be executed by the XML Interface.
output_url(*)	The output of an export request is put in this URL, instead of the default OUT directory

error	Automatically created if an import fails. The XML interface fills this property with the reason for the failure.

Note for the properties marked with (*):

These properties are specified when using the XML interface via Client Event Manager.

3 Import/Export Data

You can trigger data import/export in two ways:

- Using the **Configuration** or **Analyzer** application
- Using the **Client Event Manager**.

Both ways require creating an XML file. This file is then saved in the *IN* folder of the interface.

3.1 XML Import/Export Configuration Dialog Box

The user can configure the properties in relation to import/export operations via the **XML import/export configuration** dialog box. This feature aids the user in defining the options and properties of the XML file without having to open the file for manual editing.

To open this dialog box, select the menu command **View > Options > XML Settings** in the **Analyzer** and **Configuration** application.

The dialog box that is displayed contains 2 tabs:

- Import properties
- Export properties.

Via these tabs, the user can set the values to be assigned to the Import and Export options, respectively: the options are those provided by default inside the database.

Keep in mind that the options affected by the settings that are applied will be those on the *Property node* level in the PropertyList: therefore, they will apply to the entire XML import/export file.

3.2 Data Export

3.2.1 Overview

This section covers the following topics:

Topic	See Section
Triggering Data Export	3.2.2
Available Functions for Exporting Data	3.2.3
Using GetXML functions	3.2.4
Using SelectXML functions	3.2.5
Using GetXMLTableData	3.2.6

3.2.2 Triggering Data Export

You can trigger Data Export in two ways:

- After selecting a task in the **Configuration** or **Analyzer** application, select the menu command **"*Object type list*" > Export > As XML**
- Put an XML file containing *a GetXML<objectType>* or a *SelectXML<objectType>* function in the *IN* folder of the interface.

3.2.3 Available Functions for Exporting Data

The following functions are available:

Configuration		**'Get' Export Function**	**'Select' Export Function**
Request Type	RT	GetXMLRequestType	SelectXMLRequestType
Sample Type	ST	GetXMLSampleType	SelectXMLSampleType
Parameter Profile	PP	GetXMLParameterProfile	Not feasible.
Parameter	PR	GetXMLParameterConfig	Not feasible.
Method	MT	GetXMLMethodConfig	Not feasible.
Info Profile	IP	GetXMLInfoProfile	Not feasible.
Info Field	IE	GetXMLInfoFieldConfig	Not feasible.
Protocol	PT	GetXMLProtocol	SelectXMLProtocol
Worksheet Type	WT	GetXMLWorkSheetType	SelectXMLWorkSheetType
Operational			
Sample	SC	GetXMLSample	SelectXMLSample
Request	RQ	GetXMLRequest	SelectXMLRequest
Method	ME	GetXMLMethod	SelectXMLMethod
Study	SD	GetXMLStudy	SelectXMLStudy
WorkSheet	WS	GetXMLWorkSheet	SelectXMLWorkSheet

3.2.4 Using GetXML functions

Rules

When you put an XML export file in the **IN** folder, the following rules apply:

- The **GetXml<xxx>** function in the XML-file specifies the object to retrieve.
- The request file contains the requested object ID (and only one object ID).
- You can specify the version. If not specified, the system retrieves the current version. If there is no current version, the system retrieves the version with the highest version number.
- The exported file always contains the complete structure of the exported object (e.g. sample), which includes the main object and all child objects such as ic, ii, pg, pa, me, au, gk, etc.

Example 1

The following example of an input file will get:

- The version **0001.02** of the sample type **mySampleType**
- The current version of **mySampleType2**

```
<Unilab xmlns="xsdUnilab">
     <PropertyList>
          <prop name="userid">se</prop>
          <prop name="function">GetXmlSampleType</prop>
     </PropertyList>
     <stls>
          <ST>
               <st>mySampleType</st>
               <version>0001.02</version>
          </ST>
          <ST>
               <st>mySampleType2</st>
          </ST>
     </stls>
</Unilab>
```

Example 2

The following example of an input file will get sample **20030116-027**

```
<Unilab xmlns="xsdUnilab">
     <PropertyList>
          <prop name="userid">Kirk</prop>
          <prop name="function">GetXmlSample</prop>
     </PropertyList>
     <scls>
          <SC>
               <sc>20030116-027</sc>
          </SC>
     </scls>
</Unilab>
```

3.2.5 Using SelectXML functions

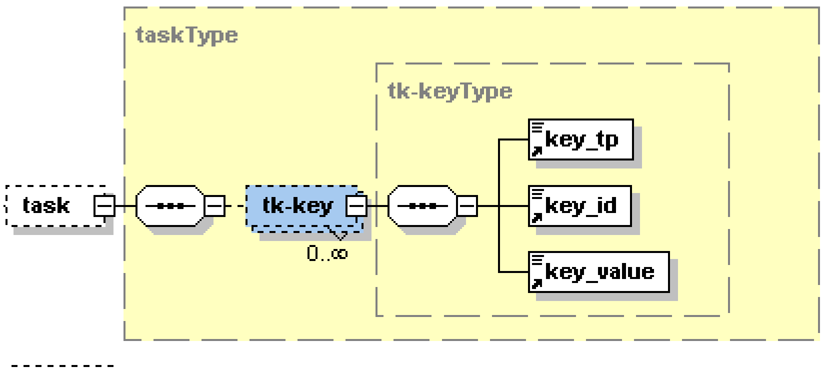
Description

The **SelectXML** functions allow the export of multiple objects with specific filter criteria. These functions use *Select<ObjectType> DB-API* calls.

Note: To be consistent with the DB-API, the filter criteria are called tasks, because they are similar to the selecting criteria used for tasks.

Diagram

Below is a general diagram of the tasks in the *xmlwrapper.xsd* file:



The tasks or filter criteria use keys with the following key types:

Key Type	Name	Description
key_tp	Key type	Indicates whether the key: • Is a group key. **Value:** e.g: “stgk”. • Contains a standard property. **Value:** [<object type>] e.g. “st”
key_id	Key identifier	Contains a group key name, or the name of a standard property.
key_value	Key value	Contains the values to be retrieved. **Note:** Wildcards are not supported.

For more detailed information regarding the available standard properties, please refer to the DB-API online help on the corresponding *Select<ObjectType> DB-API.*

Example

The example below will return all current versions of all available sample types, including all details about the used parameter profiles, parameters and methods.

Notes:

- The resulting file will be called *“Config06ImportSampleType.xml”*.
- If this file is used to import the objects in another system directly, the sample types will be created there with new major version id (since

*AllObjectsMajorV*ersion will be set to *Yes*), and object status “Approved” (since *ObjectStatus* = *‘@A’*).

```
<Unilab xmlns="xsdUnilab">
     <PropertyList>
          <prop name="function">SelectXmlSampleType</prop>
          <prop name="full_details">1</prop>
          <prop
name="output_url">Config06ImportSampleType.xml</prop>
          <prop
name="output_property">AllObjectsMajorVersion=Yes</prop>
          <prop name="output_property">ObjectStatus=@A</prop>
     </PropertyList>
     <task>
          <tk-key>
               <key_id>version_is_current</key_id>
               <key_tp>st</key_tp>
               <key_value>1</key_value>
          </tk-key>
     </task>
</Unilab>
```

3.2.6 Using GetXMLTableData

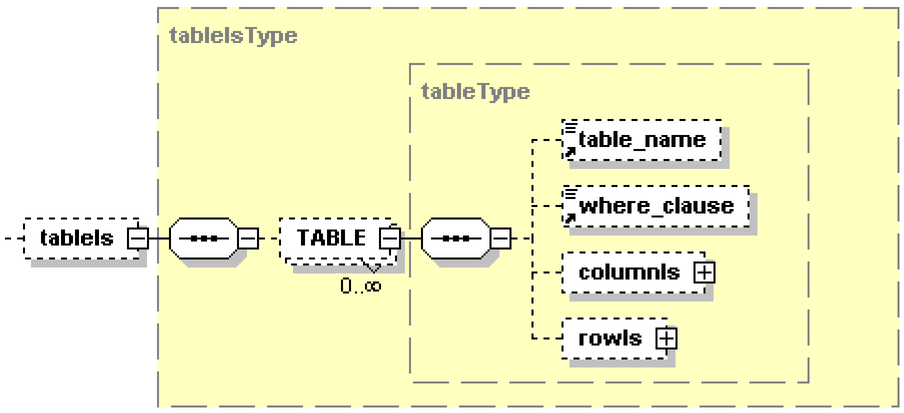
Description

The XML Interface also supports a mechanism for exporting or importing data that is not covered by DB APIs.

The GetTableData and SaveTableData APIs are used for this.

Diagram

Below is a general diagram of the tables in the *xmlwrapper.xsd* file:



Example

The example below will export:

- All data in the *uteq* table where the lab is not '-'
- The complete *utuc* table

Warning:

Do not use this fragment as an import file! For correct use, place the file in the input folder of the Client Event Manager.

```
<Unilab xmlns="xsdUnilab">
     <PropertyList>
          <prop name="function">GetXmlTableData</prop>
          <prop name="userid">se</prop>
     </PropertyList>
     <tablels>
          <TABLE>
               <table_name>uteq</table_name>
               <where_clause>where lab not in ('-
')</where_clause>
          </TABLE>
          <TABLE>
               <table_name>utuc</table_name>
          </TABLE>
     </tablels>
</Unilab>
```

3.3 Data Import

3.3.1 Overview

This Section covers the following topics:

Topic	See
Triggering Data Import	3.3.2
The Import Process	3.3.3
Rules and Tips when Importing Files	3.3.4
Import of Configuration Objects in a Unilab Database with Version Control	3.3.5
Import of Configuration Objects in a Unilab Database without Version Control	3.3.6
Import of Operational Data in a Unilab Database with 21CFR11	3.3.7
When Will XML Interface Modify Existing Data?	3.3.8

3.3.2 Triggering Data Import

You can trigger Data Import in two ways:

- After selecting a task in the **Configuration** or **Analyzer** application, select the menu command **“*Object type list*” > Import XML**
- Put an XML file in the *IN* folder of the interface.

Be sure that the files are writeable so that the files can be removed when processed.

3.3.3 The Import Process

Below is an overview of the import process:

Phase	Description	
1	The XML interface evaluates all files in the IN folder and checks whether the *Function* property in *PropertyList* either: • Is blank or does not exist, or • Has a value equal to "import" In that case, the file is identified as an import file. **Note:** All files in the IN folder that do not comply with the criteria above are ignored by the Data Import module.	
2	The XML interface attempts to import the files.	
3	**If the import was successful:**	**If the import failed:**
4	The system: • Moves the input file to the LOG folder (if logging/tracing is activated) • Moves the XSLT-transformed data file to the LOG folder (if configured) **Note:** Any warnings are recorded in a log file. By default, this file is called *unerror.log.*	The system: • Inserts the reason for failure in the input XML file. • Moves this XML file to the ERROR folder **Note:** No objects, including those that did not generate an error, are imported.

PropertyList options

When using the CEM for an XML import, the PropertyList options defined in the XML file have a higher priority compared to those defined in the XML import/export configuration dialog box.

The table below summarizes the rules that are applied for the PropertyList options when an XML file import is executed:

If the PropertyList options are configured:	The system will …
In the XML import file and the XML import/export configuration dialog box	Use those defined in the XML import file. These options will overrule that defined in the XML import/export configuration dialog box.
In the XML import file and the XML import/export configuration dialog box but some are missing in the XML import file	Merge the options of the XML import/export configuration dialog box with that of the XML import file.

3.3.4 Rules and Tips when Importing Files

Overview

The following rules and tips apply when importing files:

Rule/Tip	See Page
Including Several Objects	3-10
Object Properties	3-10
Creating or Modifying Objects	3-10
rqCreateRq and scCreateSc *options*	3-10
<Version> Node	3-11
Setting the Status	3-11
Obtaining a key-identifier via a SQL statement	3-12

Including Several Objects

It is possible to include several objects in a single file, but each file is handled within a single transaction.

Object Properties

Most of the object properties are optional, except for the primary key fields.

It is not necessary to specify all properties for an object. Simply specify the relevant ones.

Creating or Modifying Objects

The XML interface can be used to:

- Create new objects
- Modify existing object properties. Most properties can be modified. For more information, refer to the DB-API online help.

rqCreateRq and *scCreateSc* options

The *rqCreateRq* and *scCreateSc* options allow selecting between the *CreateRequest/SaveRequest* and *CreateSample/SaveSample DB-API*.

For a description of the difference in behavior of these options, refer to the DB-API online help.

<Version> Node

Next to the attributes, there are two dedicated values for the *<version>* node, which allow controlling the version handling of the imported objects.

Version	Description
NewMinor	A new minor version of the object will be created.
NewMajor	A new major version of the object will be created.

Example:

```
     <ST>
          <st>chicken meat</st>
          <version>NewMajor</version>
…..
     </ST>
```

In this example, a new major version of the sample ”chicken meat“ is created.

Setting the Status

After saving the object, the status **<ss>**, specified in the XML files, is applied by checking the actual status (after all events are processed) and performing a ***ChangeStatus***, if necessary. The same applies to the life cycle <**lc**>.

It is good practice not to specify the status or life cycles if a change status or a change life cycle is not necessary.

Note that, if you specify status @C, the object will be cancelled like in the Client application.

Example:

```
<SC>
<st>Chicken Meat</st>
     <priority>1<\priority>
<gkls>
          <GK>
               <gk_name>plant</gk_name>
               <gk_value>Brussels</gk_value>
          </GK>
</gkls>
<pgls>
          <PG>
               <pg>Chemical</pg>
               <pals>
                    <PA>
                         <pa>PCB</pa>
                         <value_f>0.001</value_f>
                    </PA>
               </pals>
          </PG>
</pgls>
</SC>
```

In this example, the system creates a sample:

- Of sample type **Chicken Meat**
- With group key **plant** set to **Brussels**
- With priority set to **1**
- With the value of the parameter **PCB** set to **0.001**.

Obtaining a key-identifier via a SQL statement

For a sample, you can specify a SQL statement to obtain the actual sample code. This can be very useful since, for example, not all equipment can provide such information. In this case, an info field or a method group key can be used to maintain the link between the ID used within the data file and the actual sample code.

Important

Using a SQL statement to obtain a key-identifier must be adopted with great caution, as it may result in performance issues (depending on the complexity of the SQL statement used).

For the following tags:

- <sc>
- <rq>
- <pa>
- <me>
- <cell>

- <ii>
- <iivalue>
- <value_f>
- <value_s>

the system automatically checks if the value starts with an equal sign.

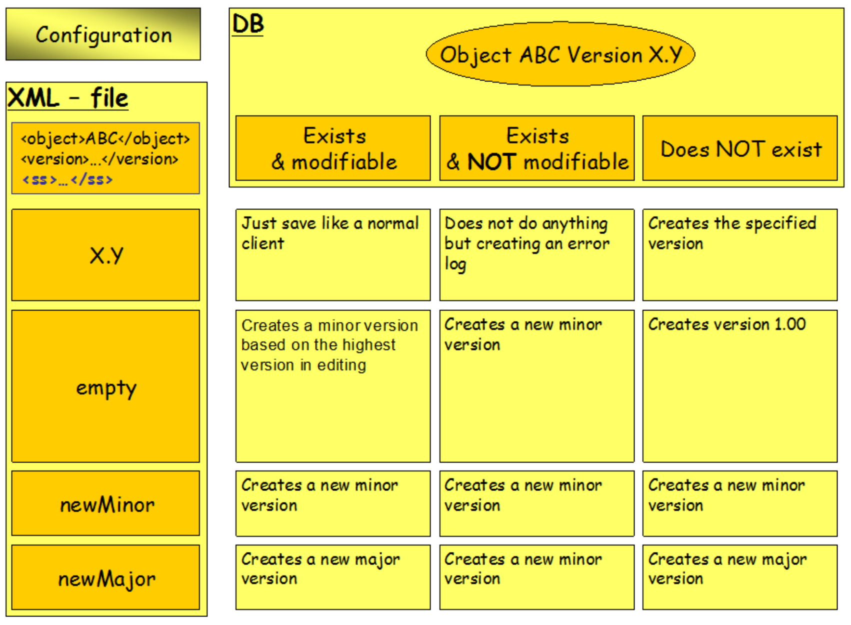
In this case, the value is first evaluated as a SQL statement that returns the actual request/sample code.

Tilde substitution in the SQL statement is possible in the SC (or RQ) context. For details in merit, please refer to *Appendix A: Tilde Substitution* of the *SIMATIC IT Unilab – Customizing the System* manual in the Unilab Product Library.

3.3.5 Import of Configuration Objects in a Unilab Database with Version Control

The illustration below provides an overview of all possible situations that can occur when an XML file containing a configuration object, is imported in a *21 CFR Part 11 enabled* database.

Note: The XML import behaves the same as when a user opens a configuration object in the regular client.



Configuration

DB

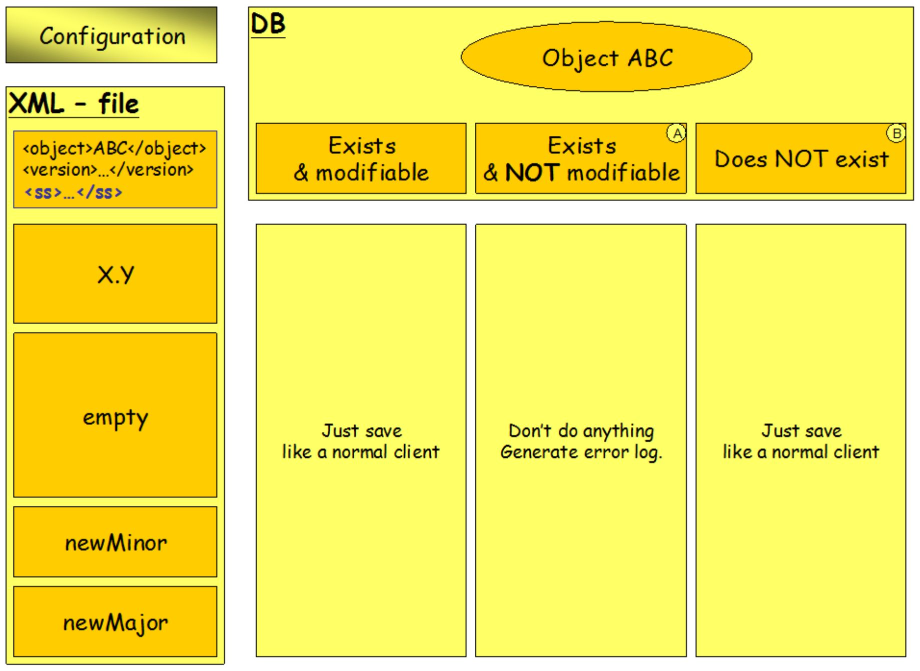
Object ABC Version X.Y

XML - file <object>ABC</object> <version>…</version> <ss>…</ss>	Exists & modifiable	Exists & NOT modifiable	Does NOT exist
X.Y	Just save like a normal client	Does not do anything but creating an error log	Creates the specified version
empty	Creates a minor version based on the highest version in editing	Creates a new minor version	Creates version 1.00
newMinor	Creates a new minor version	Creates a new minor version	Creates a new minor version
newMajor	Creates a new major version	Creates a new minor version	Creates a new major version

3.3.6 Import of Configuration Objects in a Unilab Database without Version Control

The illustration below provides an overview of all possible situations that can occur when an XML file containing a configuration object is imported in a *non-21 CFR Part 11* enabled database.

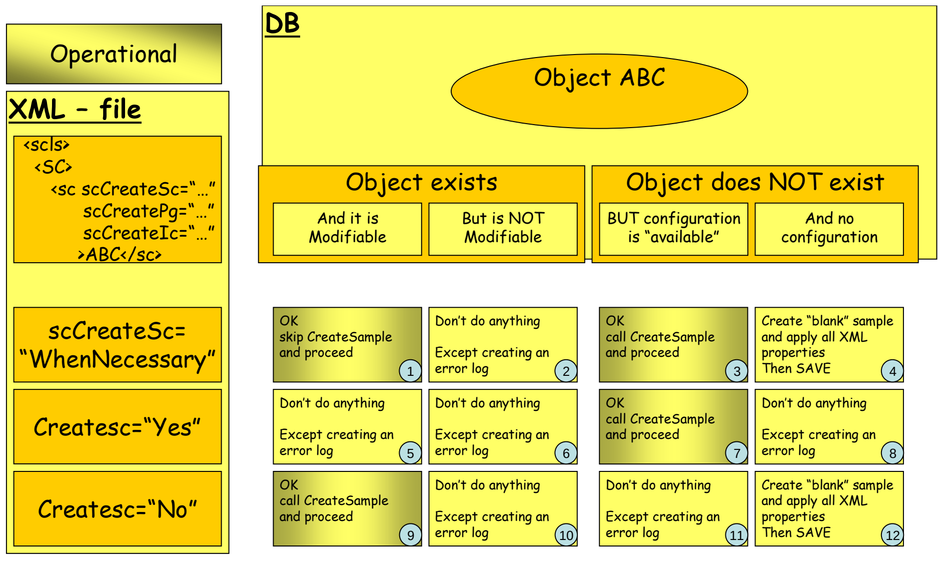
Note: The XML import behaves the same as when a user just opens a configuration object in the regular client.



Basically, the system ignores the version number.

3.3.7 Import of Operational Data in a Unilab Database with 21CFR11

The illustration below provides an overview of all possible situations that can occur when an XML file containing an operational object is imported in a *21 CFR Part 11* enabled database.


Operational
XML - file
<scls>
<SC>
<sc scCreateSc="…"
scCreatePg="…"
scCreateIc="…"
>ABC</sc>
scCreateSc= "WhenNecessary"
Createsc="Yes"
Createsc="No"
DB
Object ABC
Object exists
And it is Modifiable
But is NOT Modifiable
Object does NOT exist
BUT configuration is "available"
And no configuration
OK skip CreateSample and proceed 1
Don't do anything Except creating an error log 2
OK call CreateSample and proceed 3
Create "blank" sample and apply all XML properties Then SAVE 4
Don't do anything Except creating an error log 5
Don't do anything Except creating an error log 6
OK call CreateSample and proceed 7
Don't do anything Except creating an error log 8
OK call CreateSample and proceed 9
Don't do anything Except creating an error log 10
Don't do anything Except creating an error log 11
Create "blank" sample and apply all XML properties Then SAVE 12

3.3.8 When Will XML Interface Modify Existing Data?

The XML input file will modify an existing sample, only if:

- The sample code is specified in the file.

  If not, the sample is considered as non-existing.
- The sample type in the XML file (if specified) matches the sample type of that sample in the database.

  If not, an error will be generated and there will be no data modifications.
- The sample type version in the XML file (if specified) matches the sample type version of that sample in the database.

  If not, the system generates an error and there is no data modification.
- If *pgnode, panode, etc* matches exactly. This is necessary only if they are specified in an import request.

  If they do not match, the system creates a new object with that node ID. If this creation is not possible, an error will occur.

  Note: If *pgnode, panode, etc* are not specified, the first object with a matching ID is used. If no object with a matching ID is found, the system creates a new object.

Note: The same applies for the other operational objects.

3.4 Export to Database

The XML interface is used for the *Export to Database* functionality, provided in the **Configuration** and **Analyzer** applications.

To configure the list of export databases, you must add them in the *UTULPEERS* database table. To set a new database, select the menu command **Define >Table >Export Databases**, in the Configuration application.

3.5 Customization

3.5.1 About Customization

The *SIMATIC IT Unilab XML Interface* allows customizations to be performed with ease through the standard customization possibilities of the SIMATIC IT Unilab system.

How to modify custom functions is explained in the document **SIMATIC IT Unilab - Customizing the System** in the Unilab Product Library.

3.5.2 Examples

Example 1

When an object has been imported or modified with the XML Interface, a dedicated server event occurs:

- **ObjectImported** for configuration objects
- **SampleImported**
- **RequestImported**
- **MethodImported** for operational objects.

Example 2

The Unilab server can raise two dedicated client alerts towards a **ClientEventManager** XML Interface:

- **ExportObject**: The requested object will be exported.
- **AskImportObject**: A request to export data will be issued to another server. The requested data is placed in the **IN** folder of the XML Interface, and will be imported.

For more information about these events, see **Alert Data Details** in the document **SIMATIC IT Unilab – Client Event Manager** in the Unilab Product Library.

Example 3

The imported or exported XML can be manipulated with the following preferences:

- **ExportXslt**: If set, the given transformation is applied on the exported data, after the actual export.
- **ImportXslt**: If set, the given transformation is applied on the imported data, before the actual import.

Example 4

Full control is obtained by modifying the custom function that is executed before the import (*CF_XML_Import_Transform*), or before the export (*CF_XML_Transform*).

The steps involved in the import/export can be determined by the user via these custom functions: rather than handling the XML file’s contents as a whole (i.e. “one-shot” processing), you can determine how such contents may be “split” and processed as separate entities making up the original XML file (i.e. individual transactions). In certain situations (e.g. when managing particularly serious errors, or when the user wants to have full audit-trail details regarding those changes that a sample created outside Unilab underwent prior to its import), opting in favor of “split” processing over “one-shot” processing may be preferable.

One of the examples provided in custom function *CF_XML_Import_Transform* illustrates this concept.

4 The XML Web Service

4.1 About the XML Web Service

All functionality of the XML interface is available through a **.NET** web service.

Security Issues

XML Web Service security is guaranteed, as:

- It is handled via incoming SOAP requests.
- It uses the https protocol (with SSL – Secure Socket Layer).
- It checks the authentication of the user
- The XML data are validated with the SIMATIC IT Unilab XML scheme.
- It passes the request to the XML Interface.
- For a data export request, it returns the result as the **OUT** parameter of a synchronous call.
- For a data import request, no feedback is returned.

4.2 Added Functions

This Web Service adds two main functions to the XML Interface:

Function	Description
Availability	The service can be made available on intranet/internet.
Security	The XML interface does not offer access control in itself. Anyone with access to the interface folders can import/export data from/to the SIMATIC IT Unilab system with a Unilab user ID. Therefore, access to these folders should be restricted. However, using the XML Web Service, you can apply full security. This security is not based on the SIMATIC IT Unilab user/password combination, but is based on the (optional) use of XML signatures and certificates. Signatures and certificates are briefly explained in the remainder of this chapter.

More information: For an overview of the functions and supported objects exposed by the web service, refer to Section 1.5 Web service.

4.3 Installation and Configuration

4.3.1 Installation

Before you can work with the web service, you must install *Microsoft .NET framework.*

If certificates will be used, you also need to install *Microsoft Web Services Enhancements.*

For more information on the supported versions, see the *SIMATIC IT Unilab – Prerequisites* document in the Unilab Product Library.

4.3.2 Registry settings

Some settings are necessary for signing messages.

The *ClientEvtMgr* section contains some dedicated registry settings for the *XML Web Service*. For details regarding these registry settings, please see section **The Client Event Manager settings** in the **SIMATIC IT Unilab – Registry Settings** manual in the Unilab Product Library.

4.4 XML Signature

4.4.1 About this Section

Any XML file issued to the XML Web Service can bear an (optional) signature. An XML signature uses a hash algorithm and an asymmetric encryption algorithm. This section provides you with some information on XML Interface's use of signatures. It covers the following topics:

Topic	See
Encryption	4.4.2
Decryption	4.4.3
Algorithms	4.4.4
Nodes	4.4.5
Example	4.4.6

4.4.2 Encryption

A hash string is produced based on the data that must be signed. This hash string is then encrypted, using the private key. This encrypted hash string is the signature.

4.4.3 Decryption

The receiver of the signed message can calculate the hash of the signed data and decrypt the signature with the public key. If these two strings are equal, then the signature is valid. It proves that the data was not changed after the file was signed, and that it was signed by the owner of the private key.

4.4.4 Algorithms

The **SIMATIC IT Unilab XML Interface** uses:

- **SHA1** (Secure Hash Algorithm), as hashing algorithm. Published by NIST in the mid 1990s, it produces a 160-bit hash value.
- **RSA** as encryption algorithm. This is an asymmetric encryption algorithm. Asymmetric cryptography uses two mathematically-related keys. One key, which is made public, is used for encrypting. The other key, which must be kept secret, is used for decrypting. Anyone can access the public key and encrypt data, but only the owner of the private key can decrypt the encrypted data. RSA is the most used asymmetric cryptography algorithm.

Note: Incoming messages may have any hashing algorithm supported with *.Net*.

4.4.5 Nodes

Signed Unilab XML files will have <*UnilabSigned*> as root. This root will have two child nodes: a <*Signature*> child and a <*Unilab*> child.

4.4.6 Example

This is an example of a signed Unilab xml file:

```
<UnilabSigned>
     <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
          <SignedInfo>
               <CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
               <SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
               <Reference URI="#xpointer(/)">
                    <Transforms>
                         <Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
                    </Transforms>
                    <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

     <DigestValue>WX/A0THmQnFG1x9FTqevwsQTQNM=</DigestValue>
               </Reference>
          </SignedInfo>

     <SignatureValue>FIoHmHJrnhvk/…PMS3YB0n9vtLvM=</Signatur
eValue>
```

```
          <KeyInfo>
               <KeyValue
xmlns="http://www.w3.org/2000/09/xmldsig#">
                    <RSAKeyValue>

     <Modulus>q+…+CY9sWpjGnheEyafNk=</Modulus>
                         <Exponent>AQAB</Exponent>
                    </RSAKeyValue>
               </KeyValue>
          </KeyInfo>
     </Signature>
      <Unilab xmlns="xsdUnilab">
               Unilab data….
     </Unilab>
```

Note: The public key is part of the <*Signature*> section.

4.5 User Authentication

To guarantee the identity of the sender, the **Unilab Web Service** has to check whether the public key, passed with the signature, is really the key that belongs to the sender.

SIMATIC IT Unilab XML Web Service supports two methods for handling user authentication:

Method	See
Using a List of Registered Users	4.5.1
Using Certificates signed by a certification authority	4.5.2

4.5.1 List of Registered Users

For each user that is allowed to use the web service, a file named **<userid>.xml** must be stored in a dedicated folder (set with the **PublicKeysFolder** registry setting). This file contains the public RSA key.

Example

Below is an example of such a file:

```
<RSAKeyValue>
     <Modulus>q+adadfasdf…b9zCBKwdOzCCY9sWpjGnheEyafNk=</Mod
ulus>
     <Exponent>AQAB</Exponent>
</RSAKeyValue>
```

Process

The process is as follows:

Phase	Description
1	The system retrieves: • The user name from the propertylist/prop[@name='userid'] node • The public key from the xml signature
2	The system verifies that the public key matches the key in the *<userid>.xml* file.
3	The System checks if the user is registered. If the user is not found to be registered, the *Register web service* function can be used for its registration.
4	If there is a match on both counts, the file is approved.

The Register Web Service Function

The administrator can choose not to expose the function and maintain this list manually.

This user ID has to map with an existing user ID in Unilab. It is not possible to overwrite this public key for an already-registered user.

4.5.2 Certificates

For working with certificates, Microsoft *Web Services Enhancements (WSE) 2.0* is required. This package can be downloaded from the Microsoft website http://msdn.microsoft.com/webservices/.

The .NET way of securing web services with a Certificate is to sign the SOAP document with a certificate.

Outgoing messages

One way of digitally signing SOAP packages with a certificate is to make use of a policy file (see the ***WSE 2.0*** documentation on how to do this). The policy file can also be used for encrypting the SOAP messages.

Another way is to specify the *CertificateSubject* registry setting. Web Service will sign all outgoing SOAP messages with the certificate issued to this *CertificateSubject*.

The referred certificate should be stored in the *local machine/trust root* store.

Incoming messages

The WSE will validate the certificates of the incoming signed SOAP messages. The web service will use the name of the person to which the certificate is issued. This name can be the user ID or the actual name of a known SIMATIC IT Unilab user.

In the ***config.xml*** file (a standard configuration file for web services), you can specify whether incoming SOAP messages should be digitally signed.

Note: The private key corresponding to the public key of the certificate should be accessible for the web service. Available certificates can be viewed with *Microsoft Management Console*. To start this console, select **Start > Run** and enter **mmc**.

5 Exchanging Data between Servers

5.1 Introduction

Based on the *Unilab XML Interface***,** an easy exchange of data between different SIMATIC IT Unilab servers is possible. This exchange can be started from both the **Configuration** application (for configuration data) and the **Analyzer** application (for operation data).

The exported data is transferred to an XML-file; this file is put into the IMPORT_URL folder (refer to "*UTULPEERS columns"*).

5.2 Exchanging Data

5.2.1 What can be exported?

You can export:

- An entire list such as a request type or a sample type list.
- More specific data such as parameter profiles, parameters and methods.

Remark: This operation may take a while, depending on the amount of sample data you want to export.

5.2.2 To which databases?

The databases, to which you can export sample data, are defined in the export databases table *UTULPEERS* of the database from which you want to export data.

You can define these export databases with the *Edit Table* function.

5.3 The *Utulpeers* Database Table

5.3.1 *UTULPEERS* Export Databases Table

The export databases table named *UTULPEERS* allows you to store a number of databases to which you can export sample data. These export databases will be available in a predefined list for the **Export > To Database** command in the **Analyzer** or **Configuration** application.

5.3.2 UTULPEERS columns

The *UTULPEERS* table has the following structure:

Column name	Data Type	Description
SID	VARCHAR	The unique identification code of the database
DESCRIPTION	VARCHAR	A description of the database
IMPORT_URL	VARCHAR	The URL (path) to the database. *Note the backslash character '\' is default configured as the pre-amble character for input via bar scanner. Therefore, it is not possible to enter this character directly in the GUI; however, inserting it with copy/paste from another application works fine.*