SV Web Services Toolkit Developer's Manual

ABSTRACT:

EVOLUTIONS

Revision	Written by	Action	Date	Diffusion
0.001	BL	Initial version	05/07/2004	
1.0	BL	Release SV v8.00	05/04/2005	
1.5	BL	Release SV v8.10	02/02/2006	
1.6	BL	Beta SV v8.11 Vista support	17/10/2007	
1.61	BL	Beta SV v8.2 Access to Min and Max properties added	03/07/2008	
1.7	BL	Release SV v8.2	27/09/2008	
1.8	AK	Release SV v11.0	14/11/2013	
1.9	AK	Release SV v11.1	16/12/2014	
1.10	BL	Release SV v16.2.4	19/11/2024	

CONTENT

Scope of this document **5**

SV Web Services Toolkit concepts **6**
- Introduction 6
- Architecture 6
- Licensing 7
- XML SOAP Web Services 7
- WSDL 7
- Namespaces 8
- Compatibility 9
- Basic principles 10
- Security 17
- Data types 18

Session management service **23**
- Introduction 23
- Data types 24
- OpenSession 25
- OpenCustomSession 27
- CloseSession 29
- PingSession 31

Real time data access service **33**
- Introduction 33
- Data types 34
- Subscribe 40
- SetSubscriptionParameters 42
- SubscriptionPolledRefresh 44
- CancelSubscription 46
- GetProperties 48
- Read 50
- Write 53
- Browse 56

Alarm list access service **58**
- Introduction 58
- Data types 59
- Subscribe 64
- SetSubscriptionParameters 66
- SubscriptionPolledRefresh 68
- CancelSubscription 70
- Read 72
- GetFilters 74
- ExecuteUserAction 76

Historical data access service **78**
- Introduction 78
- Data types 79
- CreateLogRequest 88
- SetLogRequestParameters 90
- QueryLogRequest 92
- CloseLogRequest 103
- ExecuteLogRequest 105
- GetLogLists 107
- GetFilters 109
- CreateTrendRequest 111
- SetTrendRequestParameters 113
- QueryTrendRequest 115
- CloseTrendRequest 117
- ExecuteTrendRequest 119

Annex 1: Feature limitations **121**

Annex 2: WebVue Java applet - Advanced use **122**

Introduction to WebVue 122

WebVue and the OpenCustomSession method 123

Using the WebVue applet in secured environments 125

Annex 3: Properties management **127**

Scope of this document

The scope of this document is to describe and explain functional aspects of the SV *Web Services Toolkit*. It is targeted at developers willing to implement a Web Services client taking advantage of the SV *Web Services Toolkit* public interfaces.

SV Web Services Toolkit concepts

Introduction

SV is intended to be a web services server for the following accesses:

- Session management
- Real time data access
- Real time alarm access
- Historical data access: Logged events and trends

They are accessible through 4 XML SOAP web services known as the *Web Services Toolkit* and named:

- *SessionContext*
- *RealTimeData*
- *RealTimeAlarm*
- *HistoricalData*

The *Web Services Toolkit* is available since SV version 8.00. See the Compatibility section for more information about support for the different SV versions.

Architecture

The *Web Services Toolkit* is a set of 4 XML SOAP Web Services accessible over http.
SV relies on the *Microsoft Internet Information Server* (*IIS*) to manage the http layer.
The *Web Services Toolkit* is compatible with:

- *IIS 7,* shipped with *Windows Vista* and *Windows Server 2008*.
- *IIS 7.5,* shipped with *Windows 7* and *Windows Server 2008 R2*.
- *IIS 8.0,* shipped with *Windows 8* and *Windows Server 2012.*
- *IIS 8.5,* shipped with *Windows 8.1* and *Windows Server 2012 R2.*

See the SV documentation for more information about the *Web Services Toolkit* pre-requisite and *IIS* installation.

In SV architecture, one of the SCADA station can be the Web application server. This station must have the SCADA project running, and hosts *IIS*.

Licensing

The *Web Services Toolkit* is licensed. To be able to have the Web application server answering to calls, you need a dongle with at least one WebVue connection allowed. One session consumes one connection. See Session management for more information about sessions.

XML SOAP Web Services

Simple Object Access Protocol (SOAP) is a protocol for information exchange based on XML styled messages. It is a simple Request/Response protocol. The World Wide Web Consortium (W3C) is in charge of producing recommendations about the use of the protocol and its evolutions.

XML Web Services is a technique based on XML SOAP messages to exchange information in a client-server manner in a web environment. It has been designed and specified by a group of industrial partners and software editors who agreed on using proven existing web technology to insure inter operability: http, XML and SOAP. It is now evolving under the responsibility of the W3C.

Please refer to the W3C web site for complete and up to date information:

- About XML SOAP: http://www.w3.org/2000/xp/Group/
- About Web Services: http://www.w3.org/2002/ws/

WSDL

An XML SOAP Web Service interface is completely described by an XML document called WSDL (Web Services Description Language).

The WSDL is the entry point for implementing a web service client because it contains everything the client needs to know:

- Namespace of the web service.
- Specific datatypes and structures
- SOAP messages description of requests to the server.
- SOAP messages description of responses to the client.
- Binding to the underlying transport layer

The WSDL is used by developers to generate sets of datatypes and classes allowing a simple use of the web service. SOAP messages should never been built up nor interpreted “by hand”, but be dealt with thanks to an XML SOAP framework, allowing the developer working on the web service client implementation to manipulate only objects with properties and methods, and data types.

Such WSDL translation to objects is usualy called “wrapper generation”. When using Microsoft *Visual Studio*, this step is lead directly by the *Visual Studio.Net* IDE when adding a web service as “Web reference” in a project. When using a *Java* environment, popular libraries such as *Axis* are available to do so.

Once the SV *Web Services Toolkit* is deployed on a PC, you can access its WSDL files by adding "?WSDL" at the services URIs.

For example, when using a Web browser on a PC where the SV *Web Services Toolkit* is hosted, use the following URLS:

- http://localhost/SessionContext/SessionContext.asmx?WSDL to get the *SessionContext* service WSDL.
- http://localhost/RealTimeData/RealTimeData.asmx?WSDL to get the *RealTimeData* service WSDL.
- http://localhost/RealTimeAlarm/RealTimeAlarm.asmx?WSDL to get the *RealTimeAlarm* service WSDL.
- http://localhost/HistoricalData/HistoricalData.asmx?WSDL to get the *HistoricalData* service WSDL.

WSDL files are also available on SV distribution CD in the *Toolkit* section.

Namespaces

An XML namespace is a collection of names which are used in XML documents to uniquely identify XML elements and attributes.

The name of a web service is not enough to identify it because it is not a UUID. Nothing prevents 2 web service providers to name there services in the same way. For example, you may find more than one service called *wsWeatherForecast* because a lot of companies offer that kind of service and nothing prevent someone from naming his own web service as one of his competitor.
To avoid that kind of collision, a web service needs to belong to a namespace to uniquely identify it and its content (its datatypes, SOAP messages…). It is a good practice that the namespace is a URL derived from a registered domain name. Doing so insure that the couple web service name + namespace is unique.

Namespaces for the SV *Web Services Toolkit* are derived from the following domain name:

WebServicesToolkit.net

As the toolkit contains 4 separated web services, there are 4 different namespaces:

- http://www.WebServicesToolkit.net/SV/SessionContext/v1.0/
- http://www.WebServicesToolkit.net/SV/RealTimeData/v1.0/
- http://www.WebServicesToolkit.net/SV/RealTimeAlarm/v1.0/
- http://www.WebServicesToolkit.net/SV/HistoricalData/v1.0/

For each web service, you will find in its WSDL the namespace of the service, and each of the items defined in the web service (datatype, SOAP messages …) are also attached to this namespace due to the XML dependency management.

So a namespace uniquely identify:

- A web service (name)
- For a given version

Compatibility

Interoperability

The web application server where *IIS* is hosted and SV project is running is of course dependant on SV and *IIS* requirements. At the time of writing, supported operating systems are:

- Microsoft *Windows Vista*
- Microsoft *Windows Server 2008*
- Microsoft *Windows 7*
- Microsoft *Windows Server 2008 R2*
- Microsoft *Windows 8*
- Microsoft *Windows Server 2012*
- Microsoft *Windows 8.1*
- Microsoft *Windows Server 2012 R2*

Depending on the operating system edition, *IIS* might be unavailable (for example, *Windows 7 Home Basic Edition* does not allow the use of *IIS*).

As web services rely on widely adopted standards (mainly XML SOAP and http), a web service client can be implemented with any development language supporting those tehnologies and can target a wide range of hardware plateform. The implementation is easier if a web service framework is available. At the time of writing, we have had experiments or real implementations with the following frameworks:

- *C#*, *VB.Net* using *Microsoft Visual Studio.Net* along with the *.Net Framework* under *Microsoft* operating systems.
- *Java* using *Eclipse* along with *AXIS* librairies under *Microsoft* or *Linux* operating systems.

Implementations targeting the following client platforms have been tested:

- PC under *Microsoft* or *Linux* operating systems using *Visual Studio.Net* or *Java*.
- PDA devices running under *Microsoft Windows Mobile 2003* using *Visual Studio.Net*.

Future versions

The *Web Services Toolkit* will evolve in future SV versions.
Evolutions not breaking upward compatibility will be considered as minor and will be lead without changing the version mentioned in web services namespaces.
Changing the underlying server implementation of a web service or adding new messages or new datatypes might be considered as minor evolutions as they do not break compatibility.

Major evolutions will be lead with a change in the version mentioned in web services namespaces.
A major evolution is for example changing a mandatory item type in a SOAP message, or removing a message.

Basic principles

Web services take advantage of widely adopted technologies of the Internet field.
In particular, http relies on a client-server architecture used in a not-connected mode e.g. no one can assume what happens on the other side between 2 requests.
More than that, the web services specification does not define anything with regards to session nor transaction management. Web services specifications are not designed to describe persistency between client requests. At the time of writing, *W3C* groups are working on those subjects and will lead to additional specifications in the coming months.

Such considerations are not acceptable in the industrial world where security, reliability and safety are rules, and where subscription principles are widely used to avoid losing data.
So the SV *Web Services Toolkit* comes with a session layer to manage data persistency and resources allocation issues for each requesting client.

Session management

The *Web Services Toolkit* session layer gives SV the ability to:

- Manage data persistency between client requests.
- Associate resources allocated on the server side to the good client.
- Manage resources deallocation in case a client does not request anything for a long time (session time-out).

The session management layer permits a “connected mode” management and allows using the term “connected user”: Once a session is open, and as soon as the client software calls the server from time to time, the session remains valid, and the server may hold persistent data for it. Typical persistant data are its UserName and the language it wishes to receive the labels in, subscription data ….

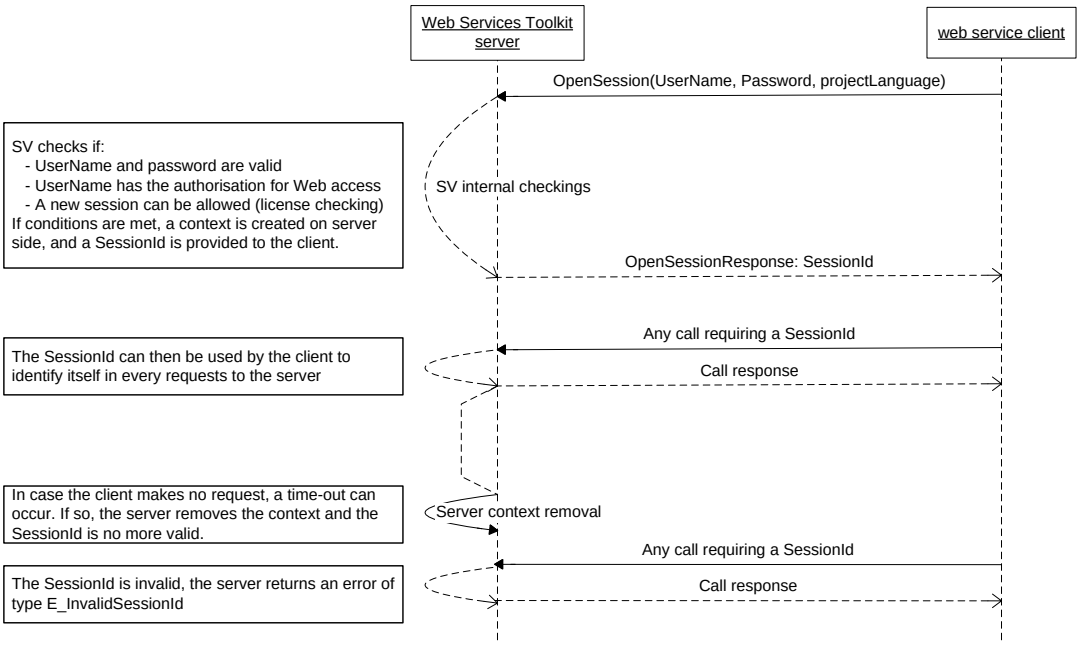


Figure 1 : Session mangement principles

Opening a session using the *OpenSession* method of the *SessionContext* web service is a mandatory operation that must be processed before any other call to the *Web Services Toolkit*.
This operation allows the SV Web application server to:

- Identify and authenticate the client by checking his rights within the SCADA application. To open a session, the client must provide a valid SV username and password.
- Allocate resources (a context) to store persistant information.
- Count simultaneous connections (the number of opened sessions) with regards to the SV license.

If checkings are successful a *SessionId* is sent back to the client. This *sessionId* is to be provided in every other request to the server.

If no request is received from a given client during a certain time period, allocated resources are freed on the server side, and the *SessionId* is no longer valid. Any further request using such a "closed context Id" will lead to the following error returned to the client: *E_InvalidSessionId*.

The time before deallocation of session context is defined by the *Client session time-out* defined in the *WebVue* server configuration.

See SV documentation for more information about *WebVue* and the *Web Services Toolkit* configuration.

Error management

For every request from client software, the server answers by returning a result structure that can be used by the client for any further processing. It is a good practice for the client software developer to use a Try-Catch structure each time a call to a web service is done. Data are always returned through *out* parameters.

In addition, some methods of the *Web Services Toolkit* return multiple data. In that case, not only global return code structure is sent back by the server, but a result code is also available for each single data item returned. It allows the server to provide the client with a specific result code for every single data item requested by the client.

For example, the *Read* method of the *RealTimeData* service is called by passing a *SessionId* and an array of variable names to read (in a *VariableRequestParameters* object). It returns a global return code structure as well as an array of variable data containing a return code structure for each requested variable.

The global return code should be used by the client to check if the operation is successful, failed, or only partially successful. The *SessionId* might be invalid, leading to a code explaining this invalidity, and then no data are returned. Even if the *SessionId* is valid, that kind of request might be only partially successful because one or more of the requested variable do not exist on server side. In that case, the exception says *E_IncompleteOperation* and the result code contained in each item of the returned array gives an information specific to the data item, giving the ability to the client to guess whether the data is meaningfull or not. The result code might be *E_Succeded* if the variable exists and was read successfully or *E_NoSuchVariable* if the variable does not exist and could not be read.

If *IIS* maximum number of http connections is reached a SOAP server exception will be triggered and returned to the client.

Data timestamping

Data timestamps returned by the SV *Web Services Toolkit* methods or used as parameters are always of type *DateTime* and expressed in UTC.

Simple access to real time variables

Through the *RealTimeData* service, it is possible to make the following simple accesses to real time variables:

- Read.
- Write.
- Read properties.

Those accesses are based on a one step request/response principle:

To read one or more variable values, the client only has to call the *Read* method, providing a *SessionId* and an array of variable names. If the *SessionId* is valid, the server answers returning a structure including an array of *VariableValue* (the last known Value, Timestamp and Quality and a result code for each array item).

To write one or more variable values, the client only has to call the *Write* method, providing a *SessionId* and an array of variable names and values. If the *SessionId* is valid, the server answers returning a structure including an array of result codes (one for each variable write).

To read one or more variable properties, the client only has to call the *GetProperties* method, providing a *SessionId*, an array of variable names, and an array of properties identifiers. If the *SessionId* is valid, the server answers by returning an array of *VariableProperties*. For each variable, a detailed result code is provided.

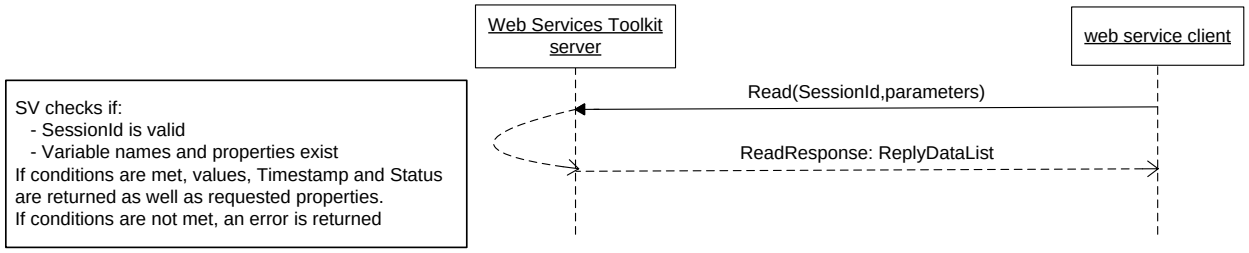


Figure 2 : Simple access to real time variables principles – Read VTQ

Subscription to real time data

The *RealTimeData* service also allows subscription to variable values.

The subscription mechanism can be divided into 3 main steps:
Step 1:
As a preliminary operation, the web service client has to call the *Subscribe* method passing the server a list of variable names to subscribe to. On receiving this request, the server allocates resources to be able to store variable values and properties each time one of the requested variable changes.
Step 2:
When necessary for client side processing, the client can call the method *SubscriptionPolledRefresh* to get the set of value changes stored on server side since the last call (or since the subscription for the first call to *SubscriptionPolledRefresh*).
The call to *SubscriptionPolledRefresh* is usually cyclic or on user request on client side.
Step 3:
When the client does not need to retrieve values any more, the subscription can be cancelled by calling the *CancelSubscription* method. Doing so allows the server to deallocate resources and stop watching for variable changes.

At any time during the life of such a subscription, the subscription parameters can be modified by calling the method *SetSubscriptionParameters*. In particular, it is possible to change the list of variables.

Susbcription options permit to retrieve only the last change of value of each variable subscribed instead of all changes since last call to *SubscriptionPolledRefresh*.

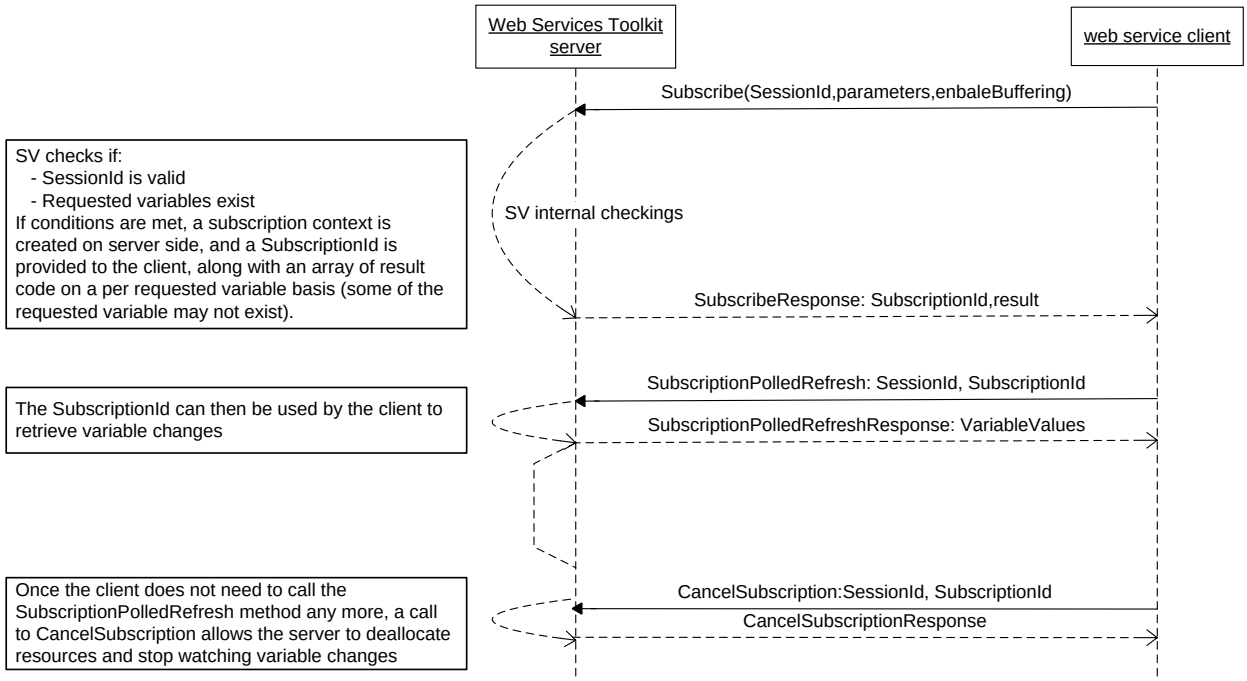


Figure 3 : Real time data subscription principles

Subscription to real time alarms

The *RealTimeAlarm* service allows subscription to alarms.

The subscription mechanism can be divided into 3 main steps:
Step 1:
As a preliminary operation, the web service client has to call the *Subscribe* method passing the server an alarm filter to subscribe to. On receiving this request, the server allocates resources to be able to store alarms and properties each time an alarm matching the filter changes.
Step 2:
When necessary for client side processing, the client can call the method *SubscriptionPolledRefresh* to get the set of alarm changes stored on server side since the last call (or since the subscription for the first call to *SubscriptionPolledRefresh*).
The call to *SubscriptionPolledRefresh* is usually cyclic or on user request on client side.
Step 3:
When the client does not need to retrieve alarms any more, the subscription can be cancelled by calling the *CancelSubscription* method. Doing so allows the server to deallocate resources and stop watching for alarm changes.

At any time during the life of such a subscription, the subscription parameters can be modified by calling the method *SetSubscriptionParameters*. In particular, it is possible to change the alarm filter.

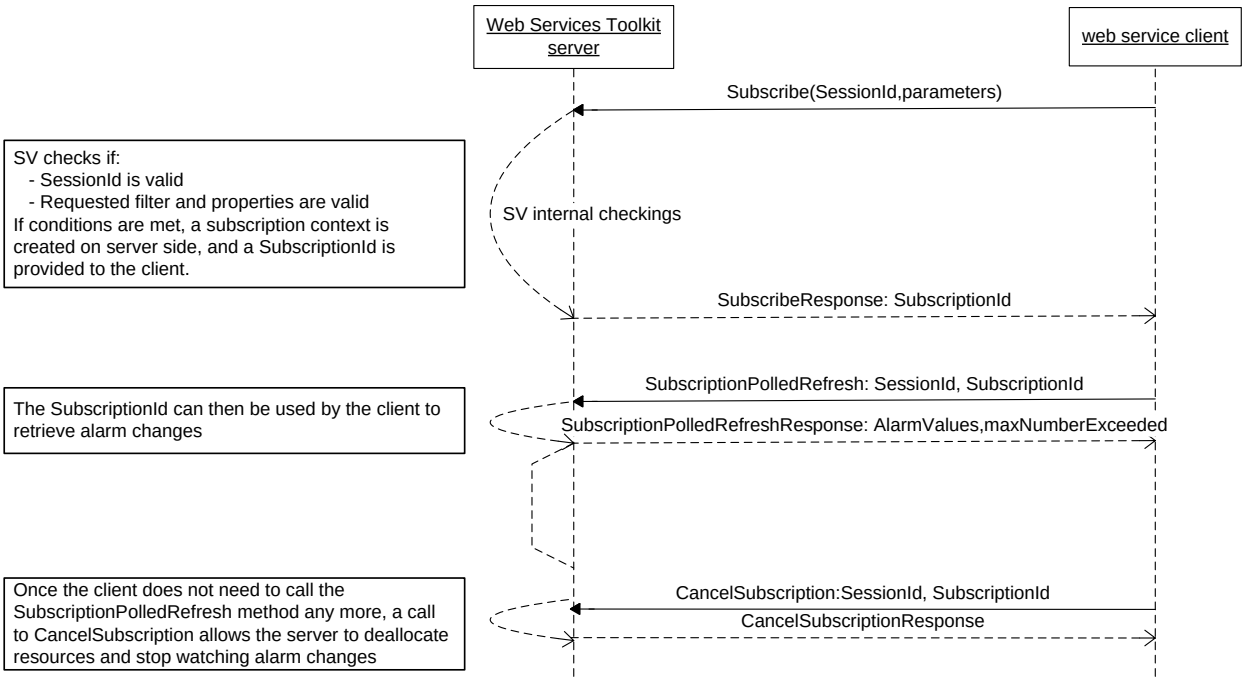


Figure 4 : Real time alarms subscription principles

Requests to historical data

The *HistoricalData* service allows requesting SV archives for logged events as well as trend data.

The log request mechanism can be divided into 3 main steps:
Step 1:
As a preliminary operation, the web service client has to call the *CreateLogRequest* method passing the server a set of parameters defining the request itself (log list name, additional filter, set of events and set of properties to retrieve in the SV archives). On receiving this request, the server allocates resources to be able to handle the request.
Step 2:
When necessary for client side processing, the client can call the method *QueryLogRequest* to get the set of logged events for a given time period.
The call to *QueryLogRequest* is usually on user request on client side.
Step 3:
When the client does not need to retrieve logged events any more, the log request can be closed by calling the *CloseLogRequest* method. Doing so allows the server to deallocate resources.

At any time during the life of such a log request, the request parameters can be modified by calling the method *SetLogRequestParameters*. In particular, it is possible to change the filter.
Each time the client calls the method *QueryLogRequest*, start time and end time are passed to the server.

This multiple steps log request handling allows designing a log viewer with features similar to the one of the SV log viewer.

In particular, it is not necessary to change the filter to change the time period requested. This helps managing time based scrolling.

It is also possible to use the *ExecuteLogRequest* method to retrieve logged events with a single call to the server. This method is usefull in case the log request is not iterated for different time periods nor for slightely changing filters.

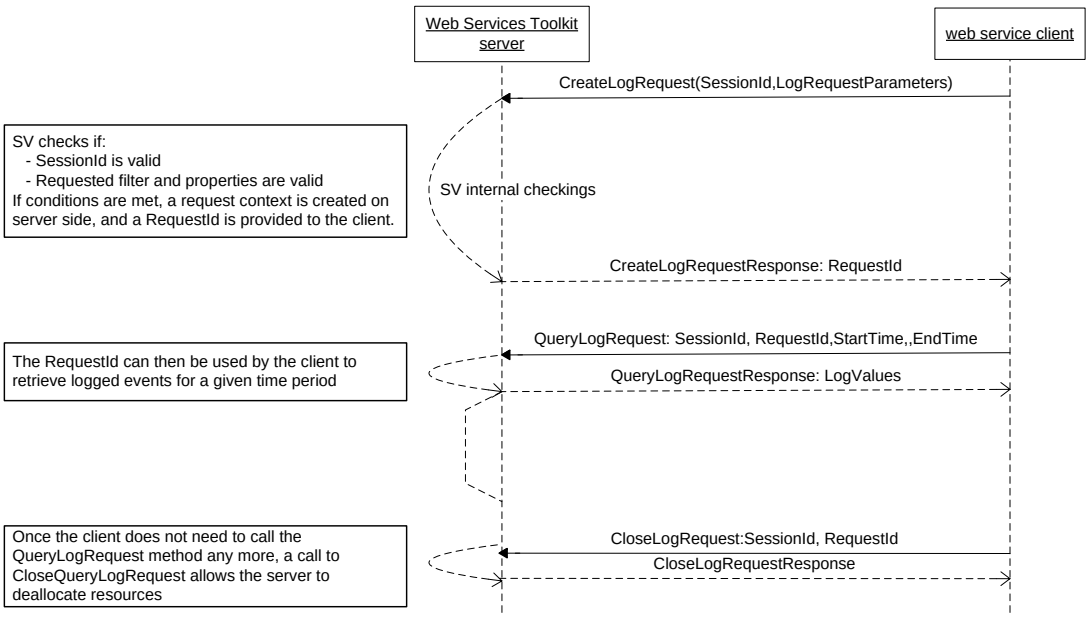


Figure 5 : Historical data request principles (a)

The trend request mechanism works in the same manner except that a variable name is passed to the server instead of a filter to define the requested data set.
The set of method available to handle trend request allows designing a trend viewer with features similar to the one of the SV trend viewer.

Predefined filters on server side

Subscribing to real time alarms as well as requesting logged events requires the definition of a filter.
The SV *Web Services Toolkit* takes advantage of filters predefined on the server side using the file *GPConf.dat*.
The *GetFilters* methods allow a Web client to get the list of filters.

See the SV documentation for more information on how to configure filters using the *GPConf.dat* file.

Bilingual SV application

The *Web Services Toolkit* is able to return localized strings if it is available on server side.
The language to use for localized strings is defined by:

- The language defined in the profile of the user (for WebVue).
- The language defined at Session opening time.
- The language defined at method calling time.

See SVLanguage for more information.
See the SV documentation for more information on how to design bilingual applications.

Security

As the XML/SOAP Web Services standards rely on already existing technologies, they do not describe anything in term of security management, in particular:

- User identification
- User authentication
- Message integrity

The *Web Services Toolkit* deals with these security aspects by existing security handling of the components involved in the communication channels.

It is mainly the identification/authentication of users at the transport layer level (http), and the identification/authentication of users at the application level (SV).

Http Client-Server security

The *Web Services Toolkit* takes advantage of the *IIS* security configuration.
In particular, *IIS* and *IIS* virtual directories can be configured for:

- No restriction (anonymous) - http requests are impersonated at the *IIS* level. It is the default configuration for virtual directories configured using SV. In that case, any client PC able to reach *IIS* can send a request to the *Web Services Toolkit*.
- Restriction to users of a given domain name.
- Restriction to client PCs belonging to a given set/range of IP addresses.

The *Web Services Toolkit* has been validated with *SSL v2* configuration of *IIS* (https). The use of *SSL v2* permits the authentication of the server from the client point of view, and http requests encryption. It is the first step to assume that messages interception does not have major consequences.

See the *IIS* documentation for more information on how to secure your http server.

SV security

SV security concerns the identification and the authentification at the SV application level.
It relies on the SV user configuration and the *Web Services Toolkit* session management. It allows SV to process client requests in a given SV user context. The user of a client application taking advantage of the *Web Services Toolkit* will be able to access data only in the limits of the user rights attached to the profile of the user given at the session opening time.
For example, if a user does not have the rights to acknowledge alarms of a given level, a call to the *ExecuteUserAction* method for an alarm belonging to one of these levels will lead to an error of type *E_InsufficientRights*.
The whole set of user rights defined in profiles applies to operations lead using the *Web Services Toolkit*:

- Variable write.
- Alarm actions (acknowledgement, masking …)

The population feature of SV does not apply to the *Web Services Toolkit* sessions.

Data types

Those types are defined in various SV web services, allowing a client to manage part of the data in a common way, whatever the SV web service used.
Be careful to the fact that unique names of those datatypes vary depending on the SV web service because the namespace is different.

ResultCode

```
<s:simpleType name="ResultCode">
  <s:restriction base="s:string">
    <s:enumeration value="E_Fail" />
    <s:enumeration value="S_Ok" />
    <s:enumeration value="E_UnknownUser" />
    <s:enumeration value="E_TooManyUsers" />
    <s:enumeration value="E_AccessDenied" />
    <s:enumeration value="E_InsufficientRights" />
    <s:enumeration value="E_InvalidSessionId" />
    <s:enumeration value="E_InvalidId" />
    <s:enumeration value="E_UnknownVariable" />
    <s:enumeration value="E_WrongParameter" />
    <s:enumeration value="E_IncompleteOperation" />
    <s:enumeration value="E_MaxExceeded" />
    <s:enumeration value="E_SvNotAccessible" />
  </s:restriction>
</s:simpleType>
```

Name	Type	Description
E_Fail	Enum value	Operation failed.
S_Ok	Enum value	Operation succeeded.
E_UnknownUser	Enum value	The user is unknown. Possible reasons are: - Wrong user name - Wrong password
E_TooManyUsers	Enum value	Server limit has been reached. The SV license does not allow more users to connect.
E_AccessDenied	Enum value	The access is denied. Possible reason is that the user does not have WebVue access granted.
E_InsufficientRights	Enum value	Insufficient rights to process the operation. Possible reasons are: - The user does not have enough rights to process a variable write. - The user does not have enough rights to process an alarm user action (acknowledgement, mask, unmask, set/reset take into account).
E_InvalidSessionId	Enum value	The Session Id is null or invalid. Possible reasons are: - The session Id is really wrong. - The corresponding session timed-out and the SV server has already removed its context.
E_InvalidId	Enum value	An Id is null or invalid (SubscriptionId or RequestId).
E_UnknownVariable	Enum value	Unknown Variable when calling a method using variable names as parameters.
E_WrongParameter	Enum value	Invalid method parameter (non-specific error).
E_IncompleteOperation	Enum value	Some operations have not been completed successfully (non-specific success). In that case, detailed errors are available item by item in the returned data.
E_MaxExceeded	Enum value	Feature limitation 1: *ResultCode*: E_*MaxExceeded* is not implemented. No method returns this error code.
E_SvNotAccessible	Enum value	Indicates that the SV is not accessible at all (i.e. it does not respond to requests for opening or closing a session or on ping). Possible reasons are:

Name	Type	Description
		- SV is not launched. - The PropertyServer is disabled on this station.

Result codes starting with an E are considered as errors.
Result codes starting with an S are successful operations.

Result

```
<s:complexType name="Result">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="code" type="tns:ResultCode" />
    <s:element minOccurs="0" maxOccurs="1" name="Description" type="s:string" />
  </s:sequence>
</s:complexType>
```

It is the result management structure returned at every method invocation.

Name	Type	Description
code	ResultCode	The result code for the request.
Description	string	Reserved. Always *null* in this version of the *Web Services Toolkit*.

SVLanguage

```
<s:simpleType name="SVLanguage">
  <s:restriction base="s:string">
    <s:enumeration value="ContextDefault" />
    <s:enumeration value="BaseLanguage" />
    <s:enumeration value="AlternativeLanguage" />
  </s:restriction>
</s:simpleType>
```

Name	Type	Description
ContextDefault	Enum Value	Value to be used to mention the default language in the context of the call. For example, if the value ContextDefault is used when opening a session, the user profile configuration will be used. See message descriptions in the following sections for more information.
BaseLanguage	Enum Value	Value to be used to override the context and force the use of the base language.
AlternativeLanguage	Enum Value	Value to be used to override the context and force the use of the alternative language.

Filter

```
<s:complexType name="Filter">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Name" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="Description" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="Expression" type="s:string" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
Name	string	Filter Name
Description	string	Filter description
Expression	string	Filter Expression

Quality

```
<s:simpleType name="Quality">
  <s:restriction base="s:string">
    <s:enumeration value="Good" />
    <s:enumeration value="Bad" />
    <s:enumeration value="BadOutOfService" />
    <s:enumeration value="GoodLocalOverride" />
    <s:enumeration value="BadCommFailure" />
    <s:enumeration value="Uncertain" />
    <s:enumeration value="UncertainEUHighLimitExceeded" />
    <s:enumeration value="UncertainEULowLimitExceeded" />
  </s:restriction>
</s:simpleType>
```

Quality represents the enumeration of the different status of a variable.

Name	Type	Description
Good	Enum Value	Status is valid (non-specific good quality).
GoodLocalOverride	Enum Value	Status is good because the variable is simulated.
Bad	Enum Value	Status is invalid (non-specific bad quality).
BadCommFailure	Enum Value	Status is bad because of a communication failure.
BadOutOfService	Enum Value	Sataus is bad because the variable is inhibited.
Uncertain	Enum Value	Status is uncertain (non-specific uncertain quality).
UncertainEUHighLimitExceeded	Enum Value	Status is uncertain because the variable value exceeds the high limit configured.
UncertainEULowLimitExceeded	Enum Value	Status is uncertain because the variable value exceeds the low limit configured.

Note:

Quality *Good*, *Bad* and *Uncertain* are non-specific status. If more information is available on server side, the specific status is used (for example *Good_LocalOverride* or *Bad_CommFailure*).

MessageHeader

```
<s:complexType name="MessageHeader">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="MessageTimeStamp" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="MessageIssuer" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="Signature" type="s:string" />
  </s:sequence>
</s:complexType>
```

The *MessageHeader* is a reserved item. It should not be used in any case.Using such elements in requests to the SV Application server may lead to incorrect results.

Session management service

Introduction

Accessing to SV data requires that a session is opened between the client and the server.

The following operations are possible on a session:

- Session opening
- Session closing
- Ping session

Data types

SessionParameters

```
<s:complexType name="SessionParameters">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="InitialWindow" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="InitialWindowBranch" type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="projectLanguage" type="tns:SVLanguage" />
    <s:element minOccurs="0" maxOccurs="1" name="AlarmFilter" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="LogFilter" type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="BeepOnNewAlarm" type="s:boolean" />
     <s:element minOccurs="1" maxOccurs="1" name="EnableStorageAsLoginLogout" nillable="true"
 type="s:boolean" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
InitialWindow	string	The name of the mimic to open when WebVue loads.
InitialWindowBranch	string	The branch of the initial window.
projectLanguage	SVLanguage	Project language associated to the session for returned localized strings. If set to ContextDefault, the language configured for the profile of the user is used. If set to BaseLanguage, the first language is used. If set to AlternativeLanguage, the alternative language is used.
AlarmFilter	string	The user default alarm filter.
LogFilter	string	The user default log filter.
BeepOnNewAlarm	boolean	The user option for the Beep on new alarm function.
EnableStorageAsLoginLogout	boolean	If set to true, the session opening and session closing will be considered as user login and logout and stored accordingly if such user actions are archived on the SV side. Default value is True for compatibility reason.

A parameter of type *SessionParameters* is passed when using the *OpenCustomSession* method. It allows overriding user profile configuration for WebVue.

See OpenCustomSession and Annex 2 for more information.

OpenSession

OpenSession

```
<s:element name="OpenSession">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="UserName" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="Password" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="projectLanguage" type="tns:SVLanguage" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

OpenSession is the container of information that represents the *OpenSession* request.

This method provides the ability to open a new session on the server. It allocates a new *SessionId* to the requester, and defines the language context.

Parameters

Name	Type	Description
UserName	string	A valid SV username
Password	string	User's password
projectLanguage	SVLanguage	Project language associated to the session for returned localized strings. If set to ContextDefault, the language configured for the profile of the user is used. If set to BaseLanguage, the first language is used. If set to AlternativeLanguage, the alternative language is used.

Comments

Except the project language, the rest of the user context is retrieved from the user profile, in particular user rights.

OpenSessionResponse

```
<s:element name="OpenSessionResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="OpenSessionResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

OpenSessionResponse is the container of information that represents the *OpenSession* response.

Parameters

Name	Type	Description
OpenSessionResult	Result	Required element. See abnormal result codes for more information.
SessionId	string	The new session Id

Comments

Abnormal result codes

Name	Description
E_TooManyUsers	See ResultCode
E_UnknownUser	See ResultCode
E_AccessDenied	See ResultCode
E_SvNotAccessible	See ResultCode

OpenCustomSession

OpenCustomSession

```
<s:element name="OpenCustomSession">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="UserName" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="Password" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="parameters" type="tns:SessionParameters" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

OpenCustomSession is the container of information that represents the *OpenCustomSession* request.

This method provides the ability to open a new session on the server with some extra parameters. It not only allocates a new *SessionId* to the requester but also allow overriding user profile web configuration (the WebVue tab of the profile configuration). The non-web items of the profile apply including user rights.

Parameters

Name	Type	Description
UserName	string	A valid SV username
Password	string	User's password
parameters	SessionParameters	Additional parameters to override user profile for WebVue.

Comments

This method has been designed to allow third-party software to create a new session prior to use it as a WebVue *SessionId* (passed as parameter to the WebVue applet).

This allows overriding User profiles configuration for specific WebVue usage. For example, it is possible for third-party softwares to override the initial window.

If such a *SessionId* is passed as parameter to the WebVue applet, it is to notice that the session will be automaticaly closed when closing the applet if it is instantiated.

See Annex 2 for more information.

OpenCustomSessionResponse

```
<s:element name="OpenCustomSessionResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="OpenCustomSessionResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

OpenCustomSessionResponse is the container of information that represents the *OpenCustomSession* response.

Parameters

Name	Type	Description
OpenCustomSessionResult	Result	Required element. See abnormal result codes for more information.
SessionId	string	The new session Id.

Comments

Abnormal result codes

Name	Description
E_TooManyUsers	See ResultCode
E_UnknownUser	See ResultCode
E_AccessDenied	See ResultCode
E_WrongParameter	See ResultCode
E_SvNotAccessible	See ResultCode

CloseSession

CloseSession

```
<s:element name="CloseSession">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

CloseSession is the container of information that represents the *CloseSession* request.

This method provides the ability to close an existing session on the server. This service clears any context on the server side.

After a call to this service, the *SessionId* is no longer valid.

Parameters

Name	Type	Description
SessionId	string	Session identifier

Comments

This method can be used to close a session openned using the *OpenSession* or the *OpenCustomSession* methods.

Forgetting to close sessions lead to a situation where new sessions can not be opened any more due to SV licence limitations.

CloseSessionResponse

```
<s:element name="CloseSessionResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="CloseSessionResult" type="tns:Result" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

CloseSessionResponse is the container of information that represents the *CloseSession* response.

Parameters

Name	Type	Description
CloseSessionResult	Result	Required element. See abnormal result codes for more information.

Comments

Abnormal result codes

Name	Description
E_InvalidSessionId	See ResultCode
E_SvNotAccessible	See ResultCode

PingSession

PingSession

```
<s:element name="PingSession">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

PingSession is the container of information that represents the *PingSession* request.

This method provides the ability for a client to make a NOOP-like call to keep a session alive without neither requesting nor retrieving data.

Parameters

Name	Type	Description
SessionId	string	A valid session Id

Comments

PingSessionResponse

```
<s:element name="PingSessionResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="PingSessionResult" type="tns:Result" />
      <s:element minOccurs="1" maxOccurs="1" name="ServerUTCTime" type="s:dateTime" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

PingSessionResponse is the container of information that represents the *PingSession* response.

Parameters

Name	Type	Description
PingSessionResult	Result	Required element. See abnormal result codes for more information.
ServerUTCTime	datetime	The SV server UTC time.

Comments

Name	Description
E_InvalidSessionId	See ResultCode
E_SvNotAccessible	See ResultCode

Real time data access service

Introduction

This service allows getting data in relation with SV real time variables.

The following operations are possible:

- Subscribe to a variable list
- Modify subscription parameters
- Get subscribed variable values
- Unsubscribe from a variable list
- Get one or more variable properties
- Read one or more variable values
- Write one or more variable values

A *SessionId* must be requested prior to using this set of methods.

Data types

VariableValue

```
<s:complexType name="VariableValue">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="result" type="tns:Result" />
    <s:element minOccurs="0" maxOccurs="1" name="value" />
    <s:element minOccurs="1" maxOccurs="1" name="Timestamp" type="s:dateTime" />
    <s:element minOccurs="1" maxOccurs="1" name="quality" type="tns:Quality" />
    <s:element minOccurs="0" maxOccurs="1" name="properties" type="tns:ArrayOfAnyType" />
    <s:element minOccurs="1" maxOccurs="1" name="IsReadOnly" type="s:boolean" />
    <s:element minOccurs="1" maxOccurs="1" name="QualityValue" type="s:int" />
    <s:element minOccurs="0" maxOccurs="1" name="alarm" type="tns:Alarm" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
result	Result	Operation Result for this particular variable.
value	object	Variable value.
Timestamp	datetime	Timestamp of variable last change.
quality	Quality	Quality of variable last change.
properties	ArrayOfAnyType	Array of requested variable properties.
IsReadOnly	boolean	Indicates whether the variable may be written or not.
QualityValue	int	Raw quality value as used internally by the SV.
alarm	Alarm	Includes additional information about the alarm status of the variable in case that the variable is of type Alarm.

This structure is used whatever the variable type (Boolean, Register or Text), so the value item is of type *object* to allow datatype casting on client side using usual cast operators.

VariableValues

```
<s:complexType name="VariableValues">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="result" type="tns:Result" />
    <s:element minOccurs="1" maxOccurs="1" name="Index" type="s:int" />
    <s:element minOccurs="0" maxOccurs="1" name="values" type="tns:ArrayOfVariableValue" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
result	Result	Operation Result for this particular variable.
Index	int	Index in the array of variable names passed at the time of invocation (method *Read* or *Subscribe*).
values	ArrayOfVariableValue	The array of *VariableValue* objects.

VariableProperty

```
<s:simpleType name="VariableProperty">
  <s:restriction base="s:string">
    <s:enumeration value="VariableName" />
    <s:enumeration value="Description" />
    <s:enumeration value="TimestampType" />
    <s:enumeration value="AssocLabel" />
    <s:enumeration value="TextAttr01" />
    <s:enumeration value="TextAttr02" />
    <s:enumeration value="TextAttr03" />
    <s:enumeration value="TextAttr04" />
    <s:enumeration value="TextAttr05" />
    <s:enumeration value="TextAttr06" />
    <s:enumeration value="TextAttr07" />
    <s:enumeration value="TextAttr08" />
    <s:enumeration value="TextAttr09" />
    <s:enumeration value="TextAttr10" />
    <s:enumeration value="TextAttr11" />
    <s:enumeration value="TextAttr12" />
    <s:enumeration value="TextAttr13" />
    <s:enumeration value="TextAttr14" />
    <s:enumeration value="TextAttr15" />
    <s:enumeration value="TextAttr16" />
    <s:enumeration value="DeferredTextAttr03" />
    <s:enumeration value="DeferredTextAttr04" />
    <s:enumeration value="DeferredTextAttr05" />
    <s:enumeration value="DeferredTextAttr06" />
    <s:enumeration value="DeferredTextAttr07" />
    <s:enumeration value="DeferredTextAttr08" />
    <s:enumeration value="DeferredTextAttr09" />
    <s:enumeration value="DeferredTextAttr10" />
    <s:enumeration value="DeferredTextAttr11" />
    <s:enumeration value="DeferredTextAttr12" />
    <s:enumeration value="DeferredTextAttr13" />
    <s:enumeration value="DeferredTextAttr14" />
    <s:enumeration value="DeferredTextAttr15" />
    <s:enumeration value="DeferredTextAttr16" />
    <s:enumeration value="BinAttr" />
    <s:enumeration value="MinValue" />
    <s:enumeration value="MaxValue" />
    <s:enumeration value="MinControlValue" />
    <s:enumeration value="MaxControlValue" />
    <s:enumeration value="MinVariableName" />
    <s:enumeration value="MaxVariableName" />
    <s:enumeration value="MinControlVariableName" />
    <s:enumeration value="MaxControlVariableName" />
    <s:enumeration value="Unit" />
    <s:enumeration value="Format" />
    <s:enumeration value="Anim0AssocLabel" />
    <s:enumeration value="Anim1AssocLabel" />
    <s:enumeration value="Cmd0AssocLabel" />
    <s:enumeration value="Cmd1AssocLabel" />
    <s:enumeration value="Log0AssocLabel" />
    <s:enumeration value="Log1AssocLabel" />
    <s:enumeration value="FormattedValue" />
    <s:enumeration value="AlarmLevel" />
  </s:restriction>
</s:simpleType>
```

It contains property descriptors corresponding to the different properties that can be retrieved for a given variable. See Annex 3: Properties management for more information.

VariableProperties

```
<s:complexType name="VariableProperties">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="result" type="tns:Result" />
    <s:element minOccurs="0" maxOccurs="1" name="properties" type="tns:ArrayOfAnyType" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
result	Result	Operation Result for this particular variable.
properties	ArrayOfAnyType	Array of requested variable properties.

Alarm

```
<s:complexType name="Alarm">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="eventCode" type="tns:AlarmSupportedEventCode" />
    <s:element minOccurs="1" maxOccurs="1" name="AckTimestamp" type="s:datetime" />
    <s:element minOccurs="1" maxOccurs="1" name="OnTimestamp" type="s:datetime" />
    <s:element minOccurs="1" maxOccurs="1" name="OffTimestamp" type="s:datetime" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
eventCode	AlarmSupportedEventCode	Event code corresponding to the event.
AckTimestamp	datetime	Timestamp of the alarm acknowledgement.
OnTimestamp	datetime	Timestamp of the alarm occurence.
OffTimestamp	datetime	Timestamp of the alarm disappearing.

VariableType

```
<s:simpleType name="VariableType">
  <s:restriction base="s:string">
    <s:enumeration value="Any" />
    <s:enumeration value="Bit" />
    <s:enumeration value="Alarm" />
    <s:enumeration value="Register" />
    <s:enumeration value="Text" />
    <s:enumeration value="Branch" />
  </s:restriction>
</s:simpleType>
```

Name	Type	Description
Any	Enum value	Any variable type, but no branch.
Bit	Enum value	Bit type.
Alarm	Enum value	Alarm type.
Register	Enum value	Analogic type.
Text	Enum value	Text type.
Branch	Enum value	Not a variable but a branch.

VariableRequestParameters

```
<s:complexType name="VariableRequestParameters">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="variableNames" type="tns:ArrayOfString" />
    <s:element minOccurs="0" maxOccurs="1" name="variableProperties"
type="tns:ArrayOfVariableProperty" />
    <s:element minOccurs="1" maxOccurs="1" name="projectLanguage" type="tns:SVLanguage" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
variableNames	ArrayOfString	The list of variables for the request.
variableProperties	ArrayOfVariableProperty	An array containing a list of property descriptors. It is the list of properties that will be returned by the server for each variable requested. Optional, defaults to null
projectLanguage	SVLanguage	SV language for returned localized strings. If set to ContextDefault, the project language of the session is used.

VariableCollectionIterator

```
<s:complexType name="VariableCollectionIterator">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="branches" type="tns:ArrayOfString" />
    <s:element minOccurs="0" maxOccurs="1" name="VariableName" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="variableType" type="tns:ArrayOfVariableType" />
    <s:element minOccurs="1" maxOccurs="1" name="CollectionRecordSize" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="DepthLevel" type="s:int" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
branches	ArrayOfString	Branch and sub branches array used as a filter. Only a single branch can be included in this parameter. This branch can be specified as a single, dot-separated string or as an array of strings, each element specifying an individual hierarchy level.
VariableName	string	Variable name.
variableType	ArrayOfVariableType	*VariableType* for the iterator.
CollectionRecordSize	int	Maximum size of the collection returned by the iterator.
DepthLevel	int	Requested branch depth. *0* returns items from all sub branches. All other numbers represent the branch level to be returned. If the branches parameters is specified the depth level is applied relative to the specified branch. That means: depth level of 1 will return direct child items and depth level of 2 will return items whose parent branch is a direct sub branch of the branch specified in the braches parameters.

VariableCollection

```
<s:complexType name="VariableCollection">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="variableType" type="tns:VariableType" />
    <s:element minOccurs="0" maxOccurs="1" name="branches" type="tns:ArrayOfString" />
    <s:element minOccurs="0" maxOccurs="1" name="VariableName" type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="IsReadOnly" type="s:boolean" />
    <s:element minOccurs="1" maxOccurs="1" name="IsLeaf" type="s:boolean" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
variableType	VariableType	Data type of the variable.
branches	ArrayOfString	Branch and sub branches array.
VariableName	string	Variable name.
IsReadOnly	boolean	*True* if the the element is read only, *False* if it is R/W.
IsLeaf	boolean	*True* if the the element is a variable, *False* if it is a Branch.

VariableCollectionRecord

```
<s:complexType name="VariableCollectionRecord">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="variableCollectionIterator"
type="tns:VariableCollectionIterator" />
    <s:element minOccurs="0" maxOccurs="1" name="variableCollections"
type="tns:ArrayOfVariableCollection" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
variableCollectionIterator	VariableCollectionIterator	The VariableCollectionIterator object for the next Browse invocation.
variableCollections	ArrayOfVariableCollection	The Variable Collection.

WriteParameters

```
<s:complexType name="WriteParameters">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="CustomUserName" type="s:string" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
CustomUserName	string	Alternative user name to be recorded by the SV and as used in the client side application.

Subscribe

Subscribe

```
      <s:element name="Subscribe">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="parameters"
type="tns:VariableRequestParameters" />
            <s:element minOccurs="1" maxOccurs="1" name="EnableBuffering" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

Subscribe is the container of information that represents the *Subscribe* request.

This method provides the ability to create a subscription on the server side, so that the server is able to have a persistent list of variables for which the client may request changes.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
parameters	VariableRequestParameters	Contains the main parameters for a variable subscription.
EnableBuffering	boolean	A flag indicating that the server must retain every value between two *SubscriptionPolledRefresh* requests. If set to *True*, all changes since the last call to *SubscriptionPolledRefresh* are available. If set to *False*, only the last change since the last call to *SubscriptionPolledRefresh* is available

Comments

SubscribeResponse

```
<s:element name="SubscribeResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SubscribeResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="SubscriptionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="results" type="tns:ArrayOfResult" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

SubscribeResponse is the container of information that represents the *Subscribe* response.

Parameters

Name	Type	Description
SubscribeResult	Result	Required element. See abnormal result codes for more information.
SubscriptionId	string	A subscription id.
results	ArrayOfResult	Contains the result of the subscription for each variable.

Comments

If one or more variable properties are not configured for one or more variable, no error is returned by the server, but property values will be returned empty when invoking the *SubscriptionPolledRefresh* method.

Abnormal result codes

ErrorCode	Description
E_InvalidSessionId	See ResultCode
E_IncompleteOperation	See ResultCode One or more requested variable may not exist. If necessary, the out parameter *results* contains detailed information variable by variable.
E_WrongParameter	See ResultCode

SetSubscriptionParameters

SetSubscriptionparameters

```
      <s:element name="SetSubscriptionParameters">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="SubscriptionId" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="parameters"
type="tns:VariableRequestParameters" />
            <s:element minOccurs="1" maxOccurs="1" name="enableBuffering" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

SetSubscriptionParameters is the container of information that represents the *SetSubscriptionParameters* request.

This method provides the ability to modify parameters for a subscription on the server side.

Those parameters include:

- The list of variables names to subscribe to.
- The list of variable properties to be returned when retrieving variable changes.
- An option enabling the server to buffer variable changes.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
SubscriptionId	string	The Identifier of the subscription to modify.
parameters	VariableRequestParameters	See Subscribe.
EnableBuffering	boolean	See Subscribe.

Comments

SetSubscriptionParametersResponse

```
      <s:element name="SetSubscriptionParametersResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SetSubscriptionParametersResult"
type="tns:Result" />
            <s:element minOccurs="0" maxOccurs="1" name="results" type="tns:ArrayOfResult" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

SetSubscriptionParametersResponse is the container of information that represents the *SetSubscriptionParameters* response.

Parameters

Name	Type	Description
SetSubscriptionParametersResult	Result	Required element. See abnormal result codes for more information.
results	ArrayOfResult	See SubscribeResponse.

Comments

Abnormal result codes

ErrorCode	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode
E_IncompleteOperation	See ResultCode See SubscribeResponse.
E_WrongParameter	See ResultCode

SubscriptionPolledRefresh

SubscriptionPolledRefresh

```
<s:element name="SubscriptionPolledRefresh">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="SubscriptionId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

SubscriptionpolledRefresh is the container of information that represents the *SubscriptionPolledRefresh* request.

This method provides the ability to get variable changes for every variables of a subscription. For each change, the set of properties is available.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
SubscriptionId	string	A valid subscription identifier.

Comments

After a call to this method the retrieved changes are no longer available on the server side for this list. If a variable is subscribed to through more than one subscription, it may be available by requesting another subscription list.

The time period at which this method is called can vary greatly from one application to another:

- It must be shorter than the session time-out period.
- Even if the session time-out can be set to be quite a big value, it is better not to do so to avoid large memory allocation on server side (to store changes between call).
- It must not be too short to avoid server request overload.

The session time-out and the period for calls to *SubscriptionPolledRefresh* should be tunned to optimize memory allocation on server side, frequency of requests to the server, and size of the request response.

SubscriptionPolledRefreshResponse

```
      <s:element name="SubscriptionPolledRefreshResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SubscriptionPolledRefreshResult"
type="tns:Result" />
            <s:element minOccurs="0" maxOccurs="1" name="replyData" type="tns:ArrayOfVariableValues" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

SubscriptionPolledRefreshResponse is the container of information that represents the *SubscriptionPolledRefresh* response.

Parameters

Name	Type	Description
SubscriptionPolledRefreshResult	Result	Required element. See abnormal result codes for more information.
replyData	ArrayOfVariableValues	The array of data (variable changes and properties) stored on server side since the previous call to *SubscriptionPolledRefresh*.

Comments

Abnormal result codes

Error Code	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode

CancelSubscription

CancelSubscription

```
<s:element name="CancelSubscription">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="SubscriptionId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

CancelSubscription is the container of information that represents the *CancelSubscription* request.

This method provides the ability to delete a subscription on the server side, so that the server is able to release the subscription context if the client is no more willing to get those variable changes.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
SubscriptionId	string	The Identifier of the subscription to cancel.

Comments

This method must be called by a client prior to close a session for every single subscription it has created.

CancelSubscriptionResponse

```
<s:element name="CancelSubscriptionResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="CancelSubscriptionResult" type="tns:Result" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

CancelSubscriptionResponse is the container of information that represents the *CancelSubscription* response.

Parameters

Name	Type	Description
CancelSubscriptionResult	Result	Required element. See abnormal result codes for more information.

Comments

Abnormal result codes

Error Code	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode

GetProperties

GetProperties

```
      <s:element name="GetProperties">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="parameters"
type="tns:VariableRequestParameters" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

GetProperties is the container of information that represents the *GetProperties* request.

This method provides the ability to get current properties for one or more variables.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
parameters	VariableRequestParameters	Contains the main parameters to get variable properties.

Comments

GetPropertiesResponse

```
      <s:element name="GetPropertiesResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="GetPropertiesResult" type="tns:Result" />
            <s:element minOccurs="0" maxOccurs="1" name="replyData"
type="tns:ArrayOfVariableProperties" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

GetPropertiesResponse is the container of information that represents the *GetProperties* response.

Parameters

Name	Type	Description
GetPropertiesResult	Result	Required element. See abnormal result codes for more information.
replyData	ArrayOfVariableProperties	The array of data (variable properties) corresponding to the request.

Comments

If one or more variable properties are not configured for one or more variable, no error is returned by the server, but property values will be returned empty when invoking this method.

Abnormal result codes

ErrorCode	Description
E_InvalidSessionId	See ResultCode
E_IncompleteOperation	See ResultCode One or more requested variable may not exist. If necessary, the out parameter *replyData* contains detailed information variable by variable.
E_WrongParameter	See ResultCode

Read

Read

```
      <s:element name="Read">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="parameters"
type="tns:VariableRequestParameters" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

Read is the container of information that represents the *Read* request.

This method provides the ability to read one or more variables' value and properties.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
parameters	VariableRequestParameters	Contains the main parameters for a variable read.

Comments

Extended attributes of a variable can be read by concatenating the variable name with the identifier of the desired attribute.

For text attributes the variable name needs to be concatenated with .TextAttr[01-16].

Please note that TextAttr01 represents the domain of a variable and TextAttr02 represents the nature of a variable.

For deferred text attributes the variable name needs to be concatenated with .DeferredTextAttr[01-16].

For the binary attribute the variable name needs to be concatenated with .BinAttr.

Threshold values of a register variable can be read by concatenating the variable name with the identifier of the desired threshold value:

.Threshold1Value

.Threshold2Value

.Threshold3Value

.Threshold4Value

The meaning of each of these values depends on the threshold system used:

ID	ppphigh/	pphigh/	phigh/	high/	low/

	pphigh/ phigh/ high	**phigh/ high/ low**	**high/ low/ lolo**	**low/ lolo pplow**	**lolo pplow ppplow**
Threshold1Value	ppphigh	pphigh	phigh	high	low
Threshold2Value	pphigh	phigh	high	low	lolo
Threshold3Value	phigh	high	low	lolo	pplow
Threshold4Value	high	low	lolo	pplow	ppplow

ReadResponse

```
<s:element name="ReadResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="ReadResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="replyData" type="tns:ArrayOfVariableValue" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

ReadResponse is the container of information that represents the *Read* response.

Parameters

Name	Type	Description
ReadResult	Result	Required element. See abnormal result codes for more information.
replyData	ArrayOfVariableValue	The array of data (variable values and properties) corresponding to the request.

Comments

If one or more variable properties are not configured for one or more variable, no error is returned by the server, but property values will be returned empty when invoking this method.

Abnormal result codes

ErrorCode	Description
E_InvalidSessionId	See ResultCode
E_IncompleteOperation	See ResultCode One or more requested variable may not exist. If necessary, the out parameter *replyData* contains detailed information variable by variable.
E_WrongParameter	See ResultCode

Write

Write

```
<s:element name="Write">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="variableNames" type="tns:ArrayOfString" />
      <s:element minOccurs="0" maxOccurs="1" name="variableValues" type="tns:ArrayOfAnyType" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

Write is the container of information that represents the *Write* request.

This method provides the ability to write one or more variables' values.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
variableNames	ArrayOfString	Variables list.
variableValues	ArrayOfAnyType	Variable values list.
writeParameters	WriteParameters	Set of additional information to be passed with the request (optional).

Comments

Extended attributes of a variable can be written by concatenating the variable name with the identifier of the desired attribute.

For text attributes the variable name needs to be concatenated with .TextAttr[01-16].

Please note that TextAttr01 represents the domain of a variable and TextAttr02 represents the nature of a variable.

For deferred text attributes the variable name needs to be concatenated with .DeferredTextAttr[01-16].

For the binary attribute the variable name needs to be concatenated with .BinAttr.

Threshold values of a register variable can be written by concatenating the variable name with the identifier of the desired threshold value:

.Threshold1Value

.Threshold2Value

.Threshold3Value

.Threshold4Value

The meaning of each of these values depends on the threshold system used:

ID	ppphigh/	pphigh/	phigh/	high/	low/

	pphigh/ phigh/ high	**phigh/ high/ low**	**high/ low/ lolo**	**low/ lolo pplow**	**lolo pplow ppplow**
Threshold1Value	ppphigh	pphigh	phigh	high	low
Threshold2Value	pphigh	phigh	high	low	lolo
Threshold3Value	phigh	high	low	lolo	pplow
Threshold4Value	high	low	lolo	pplow	ppplow

WriteResponse

```
<s:element name="WriteResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="WriteResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="results" type="tns:ArrayOfResult" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

WriteResponse is the container of information that represents the *Write* response.

Parameters

Name	Type	Description
WriteResult	Result	Required element. See abnormal result codes for more information.
results	ArrayOfResult	Contains the result of the write operation for each variable.

Comments

Abnormal result codes

ErrorCode	Description
E_InvalidSessionId	See ResultCode
E_IncompleteOperation	See ResultCode One or more requested variable may not exist. If necessary, the out parameter *results* contains detailed information variable by variable.
E_WrongParameter	See ResultCode
E_AccessDenied	See ResultCode

Browse

Browse

```
      <s:element name="Browse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="iterator"
type="tns:VariableCollectionIterator" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

Browse is the container of information that represents the *Browse* request.

This method parses the SV variable database and returns collections of variables

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
iterator	VariableCollectionIterator	Defined the browse filter and allow recursive in-depth browsing on branches.

Comments

BrowseResponse

```
    <s:element name="BrowseResponse">
       <s:complexType>
         <s:sequence>
           <s:element minOccurs="0" maxOccurs="1" name="BrowseResult" type="tns:Result" />
           <s:element minOccurs="0" maxOccurs="1" name="replyData" type="tns:VariableCollectionRecord"
/>
         </s:sequence>
       </s:complexType>
     </s:element>
```

Description

BrowseResponse is the container of information that represents the *BrowseResponse* response.

Parameters

Name	Type	Description
BrowseResult	Result	Required element. See abnormal result codes for more information.
replyData	VariableCollectionRecord	The returned data (variable records) corresponding to the request.

Comments

An object of type *VariableCollectionRecord* contains a *VariableCollectionIterator* that can be passed as parameter to the *Browse* method to iterate browsing operation.

The browse result depends on the browsing rights of the user account. This means that exclusively those variables are returned that not only match the given filter criteria but which also have a browsing level configured for which the authenticated user has the right to access.

Abnormal result codes

ErrorCode	Description
E_InvalidSessionId	See ResultCode
E_WrongParameter	See ResultCode

Alarm list access service

Introduction

This service allows getting alarm data in relation with the SV alarm manager.

The following operations are possible:

- Subscribe to an alarm filter
- Set subscription parameters
- Request alarm data for an alarm subscription
- Unsubscribe from an alarm list
- Get alarm associated labels
- Get filter names list
- Execute user actions (acknowledgement, masking, unmasking …)

A *SessionId* must be requested prior to using this set of methods.

Data types

AlarmRequestParameters

```
<s:complexType name="AlarmRequestParameters">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="projectLanguage" type="tns:SVLanguage" />
    <s:element minOccurs="1" maxOccurs="1" name="ElementMaxNumber" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="MinLevel" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="MaxLevel" type="s:int" />
    <s:element minOccurs="0" maxOccurs="1" name="eventCodes"
type="tns:ArrayOfAlarmSupportedEventCode" />
    <s:element minOccurs="1" maxOccurs="1" name="priorityFilter" type="tns:PriorityFilter" />
    <s:element minOccurs="0" maxOccurs="1" name="properties" type="tns:ArrayOfAlarmProperty" />
    <s:element minOccurs="0" maxOccurs="1" name="filters" type="tns:ArrayOfString" />
    <s:element minOccurs="1" maxOccurs="1" name="LogicalOperator" type="s:boolean" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
projectLanguage	SVLanguage	SV language for the request. If not set, the project language of the session is used.
ElementMaxNumber	int	Max number of returned alarm events. 0 = All alarms correponding to the filter.
MinLevel	int	Alarm Min level.
MaxLevel	int	Alarm Max level.
eventCodes	ArrayOfAlarmSupportedEventCode	Array of desired event codes for the request.
priorityFilter	PriorityFilter	Priority filter.
properties	ArrayOfAlarmProperty	Array of additional properties for each returned data.
filters	ArrayOfString	Array of filter to apply.
LogicalOperator	boolean	If *True*, logical AND applies to filters. If *False*, logical OR applies to filters.

Note:

To prevent server availability issues, the number of alarms that can be returned in a single request is limited to 16 000. If the *ElementMaxNumber* parameter is set to 0 (all records) or to a value greater than 16 000, no more than 16 000 alarm events will be returned, whatever the number of alarm events matching the request.

AlarmSupportedEventCode

```
<s:simpleType name="AlarmSupportedEventCode">
  <s:restriction base="s:string">
    <s:enumeration value="AlarmOnNotAck" />
    <s:enumeration value="AlarmOffNotAck" />
    <s:enumeration value="AlarmOnAck" />
    <s:enumeration value="AlarmOffAck" />
    <s:enumeration value="AlarmUnavailable" />
    <s:enumeration value="AlarmNotAccessible" />
    <s:enumeration value="AlarmInhibited" />
    <s:enumeration value="AlarmProgramMasked" />
    <s:enumeration value="AlarmVariableMasked" />
    <s:enumeration value="AlarmUserMasked" />
    <s:enumeration value="AlarmExpressionMasked" />
    <s:enumeration value="AlarmTakenIntoAccount" />
  </s:restriction>
</s:simpleType>
```

Name	Type	Description
AlarmOnNotAck	Enum value	Event code associated to an alarm in the state on not acknowledged.
AlarmOffNotAck	Enum value	Event code associated to an alarm in the state off not acknowledged.
AlarmOnAck	Enum value	Event code associated to an alarm in the state on acknowledged.
AlarmOffAck	Enum value	Event code associated to an alarm in the state off acknowledged.
AlarmUnavailable	Enum value	Event code associated to an invalid alarm, whatever the reason of invalidity. Possible reason are: - Not Accessible - Inhibited - ProgramMasked - VariableMasked - UserMasked - ExpressionMasked - TakenIntoAccount If one wishes to get detailed invalid states, the event codes below must be used instead of *AlarmUnavailable*.
AlarmNotAccessible	Enum value	Event code associated to an alarm in the state not accessible (one of the unavailable state).
AlarmInhibited	Enum value	Event code associated to an alarm in the state inhibited (one of the unavailable states).
AlarmProgramMasked	Enum value	Event code associated to an alarm in the state masked by program (one of the unavailable state).
AlarmVariableMasked	Enum value	Event code associated to an alarm in the state masked by dependence on another variable (one of the unavailable state).
AlarmUserMasked	Enum value	Event code associated to an alarm in the state masked by user (one of the unavailable state).
AlarmExpressionMasked	Enum value	Event code associated to an alarm in the state masked by dependence on an expression (one of the unavailable state).
AlarmTakenIntoAccount	Enum value	Event code associated to an alarm in the state taken into account (one of the unavailable state).

See the SV documentation for more information on alarm states and alarm management.

Feature limitation 2 : *RealTimeAlarm–AlarmSupportedEventCodes*: The alarm state

AlarmTakenIntoAccount is not implemented on the *Web Services Toolkit* server side. This event code must not be added to filter definition when using the *Subscribe*, *SetSubscriptionParameters* or the *Read* methods.

PriorityFilter

```
<s:simpleType name="PriorityFilter">
  <s:restriction base="s:string">
    <s:enumeration value="All" />
    <s:enumeration value="Mimimum" />
    <s:enumeration value="Maximum" />
  </s:restriction>
</s:simpleType>
```

Name	Type	Description
All	Enum value	All alarms corresponding to the filter are returned.
Minimum	Enum value	Alarms corresponding to the filter are returned if they belong to the smallest level requested.
Maximum	Enum value	Alarms corresponding to the filter are returned if they belong to the highest level requested.

UserActionCode

```
<s:simpleType name="UserActionCode">
  <s:restriction base="s:string">
    <s:enumeration value="Acknowledgement" />
    <s:enumeration value="Mask" />
    <s:enumeration value="Unmask" />
    <s:enumeration value="SetTakenIntoAccount" />
    <s:enumeration value="ResetTakenIntoAccount" />
  </s:restriction>
</s:simpleType>
```

Name	Type	Description
Acknowledgement	Enum value	The acknowledge action.
Mask	Enum value	The mask action.
Unmask	Enum value	The unmask action.
SetTakenIntoAccount	Enum value	The set taken into account action.
ResetTakenIntoAccount	Enum value	The reset taken into account action.

AlarmProperty

```
<s:simpleType name="AlarmProperty">
  <s:restriction base="s:string">
    <s:enumeration value="VariableName" />
    <s:enumeration value="AssocLabel" />
    <s:enumeration value="StandardLabel" />
    <s:enumeration value="UserName" />
    <s:enumeration value="Station" />
    <s:enumeration value="AlarmLevel" />
    <s:enumeration value="Informations" />
    <s:enumeration value="LogInformations" />
    <s:enumeration value="TimestampType" />
    <s:enumeration value="Description" />
    <s:enumeration value="TextAttr01" />
    <s:enumeration value="TextAttr02" />
    <s:enumeration value="TextAttr03" />
    <s:enumeration value="TextAttr04" />
    <s:enumeration value="TextAttr05" />
    <s:enumeration value="TextAttr06" />
    <s:enumeration value="TextAttr07" />
    <s:enumeration value="TextAttr08" />
    <s:enumeration value="TextAttr09" />
    <s:enumeration value="TextAttr10" />
    <s:enumeration value="TextAttr11" />
    <s:enumeration value="TextAttr12" />
    <s:enumeration value="TextAttr13" />
    <s:enumeration value="TextAttr14" />
    <s:enumeration value="TextAttr15" />
    <s:enumeration value="TextAttr16" />
    <s:enumeration value="DeferredTextAttr03" />
    <s:enumeration value="DeferredTextAttr04" />
    <s:enumeration value="DeferredTextAttr05" />
    <s:enumeration value="DeferredTextAttr06" />
    <s:enumeration value="DeferredTextAttr07" />
    <s:enumeration value="DeferredTextAttr08" />
    <s:enumeration value="DeferredTextAttr09" />
    <s:enumeration value="DeferredTextAttr10" />
    <s:enumeration value="DeferredTextAttr11" />
    <s:enumeration value="DeferredTextAttr12" />
    <s:enumeration value="DeferredTextAttr13" />
    <s:enumeration value="DeferredTextAttr14" />
    <s:enumeration value="DeferredTextAttr15" />
    <s:enumeration value="DeferredTextAttr16" />
    <s:enumeration value="BinAttr" />
  </s:restriction>
</s:simpleType>
```

It contains property descriptors corresponding to the different properties that can be retrieved for a given alarm data. See Annex 3: Properties management for more information.

AlarmValue

```
<s:complexType name="AlarmValue">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="RemovedFromList" type="s:boolean" />
    <s:element minOccurs="1" maxOccurs="1" name="Timestamp" type="s:dateTime" />
    <s:element minOccurs="1" maxOccurs="1" name="eventCode" type="tns:AlarmSupportedEventCode" />
    <s:element minOccurs="0" maxOccurs="1" name="properties" type="tns:ArrayOfAnyType" />
    <s:element minOccurs="1" maxOccurs="1" name="EventValue" type="s:long" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
RemovedFromList	boolean	Indicator that the new alarm state does not belong any more to the filter.
Timestamp	datetime	Timestamp of the event.
eventCode	AlarmSupportedEventCode	Event code corresponding to the event.
properties	ArrayOfAnyType	Array of requested properties.
EventValue	long	Reserved.

ExecuteUserActionParameters

```
<s:complexType name="ExecuteUserActionParameters">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="CustomUserName" type="s:string" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
CustomUserName	string	Alternative user name to be recorded by the SV and as used in the client side application.

Subscribe

Subscribe

```
      <s:element name="Subscribe">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="parameters" type="tns:AlarmRequestParameters"
/>
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

Subscribe is the container of information that represents the *Subscribe* request.

This method provides the ability to create a subscription to alarms on the server side, so that the server is able to have a persistent subscription context for which the client may request alarm changes.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
parameters	AlarmRequestParameters	The alarm list characteristics (filter and additional options)

Comments

Feature limitation EventCode TakenIntoAccount applies.

SubscribeResponse

```
<s:element name="SubscribeResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SubscribeResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="SubscriptionId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

SubscribeResponse is the container of information that represents the *Subscribe* response.

Parameters

Name	Type	Description
SubscribeResult	Result	Required element. See abnormal result codes for more information.
SubscriptionId	string	A subscription id

Comments

Abnormal result codes

Name	Description
E_InvalidSessionId	See ResultCode
E_WrongParameter	See ResultCode

SetSubscriptionParameters

SetSubscriptionParameters

```
      <s:element name="SetSubscriptionParameters">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="SubscriptionId" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="parameters" type="tns:AlarmRequestParameters"
/>
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

SetSubscriptionParameters is the container of information that represents the *SetSubscriptionParameters* request.

This method provides the ability to modify parameters for a subscription to alarms on the server side, so that the server is able to have a persistent subscription to alarms for which the client may request alarm changes.

Those parameters include:

- The alarm list to subscribe to (filter and types of alarm transition)
- The list of alarm properties to be returned when retrieving alarm changes

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
SubscriptionId	string	The Identifier of the subscription.
parameters	AlarmRequestParameters	See Subscribe.

Comments

Feature limitation EventCode TakenIntoAccount applies.

SetSubscriptionParametersResponse

```
      <s:element name="SetSubscriptionParametersResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SetSubscriptionParametersResult"
type="tns:Result" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

SetSubscriptionParametersResponse is the container of information that represents the *SetSubscriptionParameters* response.

Parameters

Name	Type	Description
SetSubscriptionParametersResult	Result	Required element. See abnormal result codes for more information.

Comments

Abnormal result codes

Name	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode
E_WrongParameter	See ResultCode

SubscriptionPolledRefresh

SubscriptionPolledRefresh

```
<s:element name="SubscriptionPolledRefresh">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="SubscriptionId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

SubscriptionPolledRefresh is the container of information that represents the *SubscriptionPolledRefresh* request.

This method provides the ability to get alarm data corresponding to a subscription to alarms. For each change, the set of properties is available.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
SubscriptionId	string	The Identifier of the subscription.

Comments

After a call to this method the retrieved changes are no longer available on the server side for this list. If an alarm is subscribed to through more than one subscription, it may be available by requesting another subscription list.

The time period at which this method is called can vary greatly from one application to another:

- It must be shorter than the session time-out period.
- Even if the session time-out can be set to be quite a big value, it is better not to do so to avoid large memory allocation on server side (to store changes between call).
- It must not be too short to avoid server request overload.

The session time out and the period for calls to *SubscriptionPolledRefresh* should be tunned to optimize memory allocation on server side, frequency of requests to the server, and size of the request response.

SubscriptionPolledRefreshResponse

```
      <s:element name="SubscriptionPolledRefreshResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SubscriptionPolledRefreshResult"
type="tns:Result" />
            <s:element minOccurs="0" maxOccurs="1" name="replyData" type="tns:ArrayOfAlarmValue" />
            <s:element minOccurs="1" maxOccurs="1" name="MaxNumberExceeded" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

SubscriptionPolledRefreshResponse is the container of information that represents the *SubscriptionPolledRefresh* response.

Parameters

Name	Type	Description
SubscriptionPolledRefreshResult	Result	Required element. See abnormal result codes for more information.
replyData	ArrayOfAlarmValue	The array of data (alarm changes and properties) stored on server side since the previous call to *SubscriptionPolledRefresh*.
MaxNumberExceeded	boolean	Feature limitation 3: *RealTimeAlarm*: The *MaxNumberExceeded* flag is not implemented on server side. It is always returned *False*.

Comments

For each *AlarmValue* object found in *replyData*, the item *RemovedFromList* indicates whether the alarm state still matches the filter or not. If *True*, the alarm event is returned only to indicate that it does not belong to the filter any more. In that case, the first element in the *properties* array is the variable name, and the other items of the object must not be used (*Timestamp*, *eventCode*, *EventValue*).

Abnormal result codes

Name	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode

CancelSubscription

CancelSubscription

```
<s:element name="CancelSubscription">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="SubscriptionId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

CancelSubscription is the container of information that represents the *CancelSubscription* request.

This method provides the ability to delete a subscription to alarms on the server side, so that the server is able to release the subscription context if the client is no more willing to get those alarm data.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
SubscriptionId	string	The Identifier of the subscription to cancel.

Comments

This method must be called by a client prior to close a session for every single subscription it has created.

CancelSubscriptionResponse

```
<s:element name="CancelSubscriptionResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="CancelSubscriptionResult" type="tns:Result" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

CancelSubscriptionResponse is the container of information that represents the *CancelSubscription* response.

Parameters

Name	Type	Description
CancelSubscriptionResult	Result	Required element. See abnormal result codes for more information.

Comments

Abnormal result codes

Name	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode

Read

Read

```
      <s:element name="Read">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="parameters" type="tns:AlarmRequestParameters"
/>
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

Read is the container of information that represents the *Read* request.

This method provides the ability to read alarms on the server side.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
parameters	AlarmRequestParameters	Contains the main parameters for an alarm read.

Comments

Feature limitation EventCode TakenIntoAccount applies.

ReadResponse

```
<s:element name="ReadResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="ReadResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="replyData" type="tns:ArrayOfAlarmValue" />
      <s:element minOccurs="1" maxOccurs="1" name="MaxNumberExceeded" type="s:boolean" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

ReadResponse is the container of information that represents the *Read* response.

Parameters

Name	Type	Description
ReadResult	Result	Required element. See abnormal result codes for more information.
replyData	ArrayOfAlarmValue	The array of data (alarm data and properties) corresponding to the request.
MaxNumberExceeded	boolean	Feature limitation MaxNbExceededFlag applies.

Comments

Abnormal result codes

Name	Description
E_InvalidSessionId	See ResultCode
E_WrongParameter	See ResultCode

GetFilters

GetFilters

```
<s:element name="GetFilters">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="projectLanguage" type="tns:SVLanguage" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

GetFilters is the container of information that represents the *GetFilters* request.

This method provides the ability to retrieve the set of predefined filters managed by the server.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
projectLanguage	SVLanguage	SV language for the request. If not set, the project language of the session is used.

Comments

See Predefined filters on server side.

GetFiltersResponse

```
<s:element name="GetFiltersResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="GetFiltersResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="filters" type="tns:ArrayOfFilter" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

GetFiltersResponse is the container of information that represents the *GetFilters* response.

Parameters

Name	Type	Description
GetFiltersResult	Result	Required element. See abnormal result codes for more information.
filters	ArrayOfFilter	An array containing the filters configured on server side for alarms.

Comments

Abnormal result codes

Name	Description
E_InvalidSessionId	See ResultCode

ExecuteUserAction

ExecuteUserAction

```
<s:element name="ExecuteUserAction">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="actionCode" type="tns:UserActionCode" />
      <s:element minOccurs="0" maxOccurs="1" name="variableNames" type="tns:ArrayOfString" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

ExecuteUserAction is the container of information that represents the *ExecuteUserAction* request.

This method provides the ability to perform an action on one or more alarm. This action can be:

- Acknowledgement
- Mask
- Unmask

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
actionCode	UserActionCode	The code of the action to perform.
variableNames	ArrayOfString	One or more variable names.
executeUserActionParameters	ExecuteUserActionParameters	Set of additional information to be passed with the request (optional).

Comments

Feature limitation 4 : *RealTimeAlarm-ExecuteUserAction*: The user actions *SetTakenIntoAccount* and *ResetTakenIntoAccount* are not implemented yet. Invoking this method for one of these user action code returns *S_Ok* whilst no action is performed on server side.

ExecuteUserActionResponse

```
<s:element name="ExecuteUserActionResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="ExecuteUserActionResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="results" type="tns:ArrayOfResult" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

ExecuteUserActionResponse is the container of information that represents the *ExecuteUserAction* response.

Parameters

Name	Type	Description
ExecuteUserActionResult	Result	Required element. See abnormal result codes for more information.
results	ArrayOfResult	Contains the result of the user action operation for each variable.

Comments

Abnormal result codes

ErrorCode	Description
E_InvalidSessionId	See ResultCode
E_IncompleteOperation	See ResultCode One or more requested variable may not exist, or their alarm state do not allow the requested action. If necessary, the out parameter *results* contains detailed information variable by variable.
E_AccessDenied	See ResultCode

Historical data access service

Introduction

This service allows getting archived events in relation with the SV historical manager. Those archived data are also known as logged events.

The following operations are possible:

- Create a log request
- Request logged data for a log request
- Modify a log request
- Delete a log request
- Get log list names
- Get filter names

A *SessionId* must be requested prior to using this set of services.

This service also allows getting trend data in relation with the SV historical manager.

The following operations are possible:

- Create a trend request
- Request archived trend data for a trend request
- Modify a trend request
- Delete a trend request

A *SessionId* must be requested prior to using this set of services.

Data types

LogRequestParameters

```
<s:complexType name="LogRequestParameters">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="LogListName" type="s:string" />
    <s:element minOccurs="1" maxOccurs="1" name="ElementMaxNumber" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="projectLanguage" type="tns:SVLanguage" />
    <s:element minOccurs="0" maxOccurs="1" name="eventCodes"
type="tns:ArrayOfLogSupportedEventCode" />
    <s:element minOccurs="1" maxOccurs="1" name="MinLevel" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="MaxLevel" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="LogicalOperator" type="s:boolean" />
    <s:element minOccurs="0" maxOccurs="1" name="properties" type="tns:ArrayOfLogProperty" />
    <s:element minOccurs="0" maxOccurs="1" name="filters" type="tns:ArrayOfString" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
LogListName	string	Log list Name
ElementMaxNumber	int	Max number of returned records. 0 = All record correponding to the request.
projectLanguage	SVLanguage	SV language for the log request. If not set, the project language of the session is used.
eventCodes	ArrayOfLogSupportedEventCode	Array of desired event codes for the request.
MinLevel	int	Alarm Min level
MaxLevel	int	Alarm Max level
filters	ArrayOfString	Array of filter to apply.
LogicalOperator	boolean	If *True*, only log records matching all filters will be returned (AND operator is applied between filters). If *False*, log records matching at least one of the filters will be returned (OR operator is applied between filters).
properties	ArrayOfLogProperty	Array of additional properties for each returned record

Feature limitation 5 : *HistoricalData*: GPConf.dat filters are not implemented for HDS archive units. For HDS archive units, the *filters* item must be left empty.

LogSupportedEventCode

```
<s:simpleType name="LogSupportedEventCode">
  <s:restriction base="s:string">
    <s:enumeration value="BitChangeTo0" />
    <s:enumeration value="BitChangeTo1" />
    <s:enumeration value="BitUnavailable" />
    <s:enumeration value="AlarmOnNotAck" />
    <s:enumeration value="AlarmOffNotAck" />
    <s:enumeration value="AlarmOnAck" />
    <s:enumeration value="AlarmOffAck" />
    <s:enumeration value="AlarmUnavailable" />
    <s:enumeration value="AlarmOn" />
    <s:enumeration value="AlarmOff" />
    <s:enumeration value="AlarmNotAccessible" />
    <s:enumeration value="AlarmInhibited" />
    <s:enumeration value="AlarmProgramMasked" />
    <s:enumeration value="AlarmVariableMasked" />
    <s:enumeration value="AlarmUserMasked" />
    <s:enumeration value="AlarmExpressionMasked" />
    <s:enumeration value="UserActionSendCommand" />
    <s:enumeration value="UserActionAlarmAcknowledgement" />
    <s:enumeration value="UserActionAlarmMaskUnmask" />
    <s:enumeration value="UserActionLogonLogoff" />
    <s:enumeration value="UserActionSendProgram" />
  </s:restriction>
</s:simpleType>
```

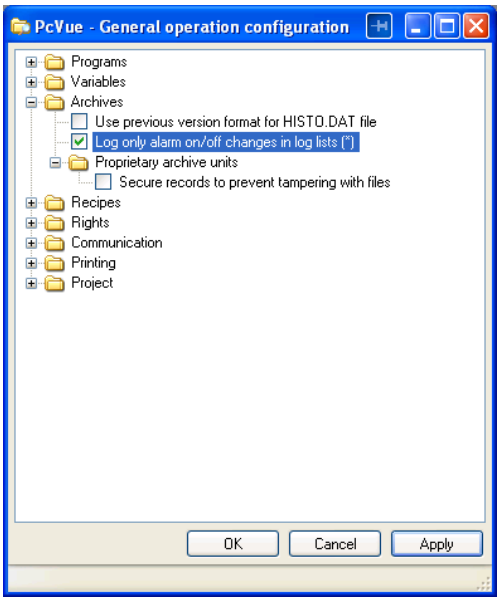
Name	Type	Description
BitChangeTo0	Enum value	Event code associated to a record for a bit variable at value 0.
BitChangeTo1	Enum value	Event code associated to a record for a bit variable at value 1.
BitUnavailable	Enum value	Event code associated to a record for an invalid bit variable.
AlarmOnNotAck	Enum value	Event code associated to a record for an alarm in the state on not acknowledged.
AlarmOffNotAck	Enum value	Event code associated to a record for an alarm in the state off not acknowledged.
AlarmOnAck	Enum value	Event code associated to a record for an alarm in the state on acknowledged.
AlarmOffAck	Enum value	Event code associated to a record for an alarm in the state off acknowledged.
AlarmUnavailable	Enum value	Event code associated to a record for an invalid alarm, whatever the reason of invalidity. Possible reason are: - Not Accessible - Inhibited - ProgramMasked - VariableMasked - UserMasked - ExpressionMasked - TakenIntoAccount If one wishes to get detailed invalid status, the event codes below must be used instead of AlarmUnavailable.
AlarmNotAccessible	Enum value	Event code associated to a record for an alarm in the state not accessible (one of the unavailable state).
AlarmInhibited	Enum value	Event code associated to a record for an alarm in the state inhibited

Name	Type	Description
		(one of the unavailable states).
AlarmProgramMasked	Enum value	Event code associated to a record for an alarm in the state masked by program (one of the unavailable state).
AlarmVariableMasked	Enum value	Event code associated to a record for an alarm in the state masked by dependence on another variable (one of the unavailable state).
AlarmUserMasked	Enum value	Event code associated to a record for an alarm in the state masked by user (one of the unavailable state).
AlarmExpressionMasked	Enum value	Event code associated to a record for an alarm in the state masked by dependence on an expression (one of the unavailable state).
AlarmOn	Enum value	Event code associated to a record for an alarm in the state on (See Note below).
AlarmOff	Enum value	Event code associated to a record for an alarm in the state off (See Note below).
UserActionSendCommand	Enum value	Event code associated to a record for a user action of type Send command.
UserActionAlarmAcknowledgement	Enum value	Event code associated to a record for a user action of type Alarm acknowledgement.
UserActionAlarmMaskUnmask	Enum value	Event code associated to a record for a user action of type Alarm masking or unmasking.
UserActionLogonLogoff	Enum value	Event code associated to a record for a user action of type Logon or Logoff.
UserActionSendProgram	Enum value	Event code associated to a record for a user action of type Send program.

Note:

Alarm state logging is widely impacted by the following option in the SV project: *Log only alarm on/off changes in log lists*:

- If it is not selected (default), the event codes *AlarmOn* and *AlarmOff* must not be used.
- If it is selected, the event codes *AlarmOnNotAck*, *AlarmOffNotAck*, *AlarmOnAck* and *AlarmOffAck* must not be used.



LogProperty

```
<s:simpleType name="LogProperty">
  <s:restriction base="s:string">
    <s:enumeration value="VariableName" />
    <s:enumeration value="Description" />
    <s:enumeration value="AssocLabel" />
    <s:enumeration value="StandardLabel" />
    <s:enumeration value="UserName" />
    <s:enumeration value="Station" />
    <s:enumeration value="AlarmLevel" />
    <s:enumeration value="Informations" />
    <s:enumeration value="LogInformations" />
    <s:enumeration value="TimestampType" />
    <s:enumeration value="TextAttr01" />
    <s:enumeration value="TextAttr02" />
    <s:enumeration value="TextAttr03" />
    <s:enumeration value="TextAttr04" />
    <s:enumeration value="TextAttr05" />
    <s:enumeration value="TextAttr06" />
    <s:enumeration value="TextAttr07" />
    <s:enumeration value="TextAttr08" />
    <s:enumeration value="TextAttr09" />
    <s:enumeration value="TextAttr10" />
    <s:enumeration value="TextAttr11" />
    <s:enumeration value="TextAttr12" />
    <s:enumeration value="TextAttr13" />
    <s:enumeration value="TextAttr14" />
    <s:enumeration value="TextAttr15" />
    <s:enumeration value="TextAttr16" />
    <s:enumeration value="DeferredTextAttr03" />
    <s:enumeration value="DeferredTextAttr04" />
    <s:enumeration value="DeferredTextAttr05" />
    <s:enumeration value="DeferredTextAttr06" />
    <s:enumeration value="DeferredTextAttr07" />
    <s:enumeration value="DeferredTextAttr08" />
    <s:enumeration value="DeferredTextAttr09" />
    <s:enumeration value="DeferredTextAttr10" />
    <s:enumeration value="DeferredTextAttr11" />
    <s:enumeration value="DeferredTextAttr12" />
    <s:enumeration value="DeferredTextAttr13" />
    <s:enumeration value="DeferredTextAttr14" />
    <s:enumeration value="DeferredTextAttr15" />
    <s:enumeration value="DeferredTextAttr16" />
    <s:enumeration value="BinAttr" />
  </s:restriction>
</s:simpleType>
```

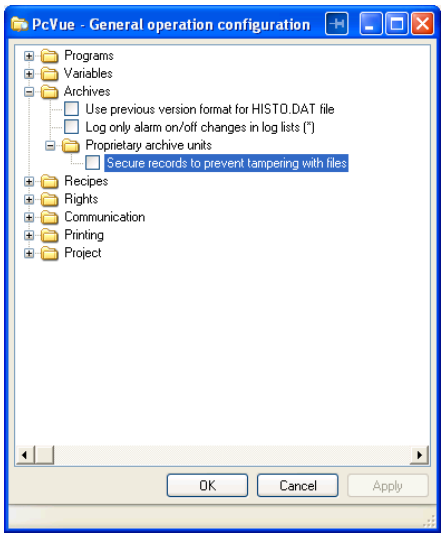
It contains property descriptors corresponding to the different properties that can be retrieved for a given log record. See Annex 3: Properties management for more information.

LogValue

```
<s:complexType name="LogValue">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="Timestamp" type="s:dateTime" />
    <s:element minOccurs="1" maxOccurs="1" name="eventCode" type="tns:LogSupportedEventCode" />
    <s:element minOccurs="1" maxOccurs="1" name="Corrupted" type="s:boolean" />
    <s:element minOccurs="1" maxOccurs="1" name="EventValue" type="s:long" />
    <s:element minOccurs="0" maxOccurs="1" name="properties" type="tns:ArrayOfAnyType" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
Timestamp	datetime	Timestamp of the event.
eventCode	LogSupportedEventCode	Event code corresponding to the event.
Corrupted	boolean	*True* if the record is detected as corrupted by the historical manager (See Note below).
EventValue	long	Reserved
properties	ArrayOfAnyType	Array of requested properties

Note:
Logged record corruption management is activated if the following option is set in the SV project: *Secure records to prevent tampering with files*:



If this option is not set, the *Corrupted* item is always returned *False*.
Corruption handling is only enabled for archive unit of type *Proprietary*. If the archive unit type is not *Proprietary*, the *Corrupted* item is always returned *False*.

LogList

```
<s:complexType name="LogList">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="Name" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="Description" type="s:string" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
Name	string	Log list name
Description	string	Log list description

TrendValue

```
<s:complexType name="TrendValue">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="value" />
    <s:element minOccurs="1" maxOccurs="1" name="Timestamp" type="s:dateTime" />
    <s:element minOccurs="1" maxOccurs="1" name="quality" type="tns:Quality" />
    <s:element minOccurs="0" maxOccurs="1" name="properties" type="tns:ArrayOfAnyType" />
  </s:sequence>
</s:complexType>
```

Name	Type	Description
value	object	Variable value.
Timestamp	datetime	Timestamp of variable change of value.
quality	Quality	Status of the variable.
properties	ArrayOfAnyType	Reserved. Must be left null (See Feature limitation).

TrendAggregateFunction

```
<s:simpleType name="TrendAggregateFunction">
  <s:restriction base="s:string">
    <s:enumeration value="Raw" />
    <s:enumeration value="WindowPixelSize" />
  </s:restriction>
</s:simpleType>
```

Name	Type	Description
Raw	Enum value	No algorithm. All trend data corresponding to the request are returned.
WindowPixelSize	Enum value	Not all data are returned, but only a subset according to a decimation algorithm based on the size of the window area where the curve will be drawn.

TrendRequestParameters

```
      <s:complexType name="TrendRequestParameters">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="VariableName" type="s:string" />
          <s:element minOccurs="1" maxOccurs="1" name="ElementMaxNumber" type="s:int" />
          <s:element minOccurs="1" maxOccurs="1" name="aggregateFunction"
type="tns:TrendAggregateFunction" />
          <s:element minOccurs="1" maxOccurs="1" name="AggregateParam1" type="s:long" />
          <s:element minOccurs="0" maxOccurs="1" name="properties" type="tns:ArrayOfTrendProperty" />
          <s:element minOccurs="1" maxOccurs="1" name="projectLanguage" type="tns:SVLanguage" />
          <s:element minOccurs="1" maxOccurs="1" name="IncludeStartBound" type="s:boolean" />
          <s:element minOccurs="1" maxOccurs="1" name="IncludeEndBound" type="s:boolean" />
        </s:sequence>
      </s:complexType>
```

Name	Type	Description
VariableName	string	Variable name.
ElementMaxNumber	int	Max number of returned trend records. 0 = All records correponding to the request.
aggregateFunction	TrendAggregateFunction	The algorithm to use to aggregate Trend data.
AggregateParam1	long	First parameters for the aggregate function. Raw: Set the parameter to 0. WindowPixelSize: Set the parameter to the width of the window where the curve will be displayed (in pixels). This decimation algorithm optimizes the amount of returned trend records according to the curve width.
properties	ArrayOfTrendProperty	Array of additional properties for each returned record.
projectLanguage	SVLanguage	SV language for the request. If not set, the project language of the session is used.
IncludeStartBound	boolean	If set to *True*, the first point found in archived data before the start date of the request is added to the returned result set.
IncludeEndBound	boolean	If set to *True*, the first point found in archived data after the end date of the request is added to the returned result set.

Note:

To prevent server availability issues, the number of records that can be returned in a single request is limited to 4000 points. If the *ElementMaxNumber* parameter is set to 0 (all records) or to a value greater than 4000, no more than 4000 points will be returned, whatever the number of records matching the request.
In the case of trend data recorded in an HDS archive unit, the points returned are chosen amoung up to 1 000 000 points found in the requested time period. If the number of points found in the time period exceeds 1 000 000, only the last 1 000 000 are processed.
In the case of trend data recorded in a proprietary archive unit, the points returned are processed on the base of the last 4000 found in the requested period.

Feature limitation 6 : *HistoricalData-TrendRequestParameters*: The *IncludeEndBound* option is only implemented for HDS archive units. Set it to the value *False* for Proprietary archive units (not implemented).
Feature limitation 7 : *HistoricalData- TrendRequestParameters*: Access to additional properties is not supported for trend requests. The *properties* item must not be used (null).

TrendProperty

```
<s:simpleType name="TrendProperty">
  <s:restriction base="s:string">
    <s:enumeration value="VariableName" />
    <s:enumeration value="Description" />
    <s:enumeration value="AssocLabel" />
    <s:enumeration value="StandardLabel" />
    <s:enumeration value="UserName" />
    <s:enumeration value="Station" />
    <s:enumeration value="AlarmLevel" />
    <s:enumeration value="Informations" />
    <s:enumeration value="LogInformations" />
    <s:enumeration value="TimestampType" />
    <s:enumeration value="TextAttr01" />
    <s:enumeration value="TextAttr02" />
    <s:enumeration value="TextAttr03" />
    <s:enumeration value="TextAttr04" />
    <s:enumeration value="TextAttr05" />
    <s:enumeration value="TextAttr06" />
    <s:enumeration value="TextAttr07" />
    <s:enumeration value="TextAttr08" />
    <s:enumeration value="TextAttr09" />
    <s:enumeration value="TextAttr10" />
    <s:enumeration value="TextAttr11" />
    <s:enumeration value="TextAttr12" />
    <s:enumeration value="TextAttr13" />
    <s:enumeration value="TextAttr14" />
    <s:enumeration value="TextAttr15" />
    <s:enumeration value="TextAttr16" />
    <s:enumeration value="DeferredTextAttr03" />
    <s:enumeration value="DeferredTextAttr04" />
    <s:enumeration value="DeferredTextAttr05" />
    <s:enumeration value="DeferredTextAttr06" />
    <s:enumeration value="DeferredTextAttr07" />
    <s:enumeration value="DeferredTextAttr08" />
    <s:enumeration value="DeferredTextAttr09" />
    <s:enumeration value="DeferredTextAttr10" />
    <s:enumeration value="DeferredTextAttr11" />
    <s:enumeration value="DeferredTextAttr12" />
    <s:enumeration value="DeferredTextAttr13" />
    <s:enumeration value="DeferredTextAttr14" />
    <s:enumeration value="DeferredTextAttr15" />
    <s:enumeration value="DeferredTextAttr16" />
    <s:enumeration value="BinAttr" />
  </s:restriction>
</s:simpleType>
```

It contains property descriptors corresponding to the different properties that can be retrieved for a given trend record. See Annex 3: Properties management for more information.

CreateLogRequest

CreateLogRequest

```
<s:element name="CreateLogRequest">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="parameters" type="tns:LogRequestParameters" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

CreateLogRequest is the container of information that represents the *CreateLogrequest* request.

This method provides the ability to create a request on the archived events on the server side, so that the server is able to have a persistent request the client may use to retrieve data for different time periods without passing a complete parameter set.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
parameters	LogRequestParameters	Log parameters for the request (filter and additional options).

Comments

CreateLogRequestResponse

```
<s:element name="CreateLogRequestResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="CreateLogRequestResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="RequestId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

LogRequestResponse is the container of information that represents the *LogRequest* response.

Parameters

Name	Type	Description
CreateLogRequestResult	Result	Required element. See abnormal result codes for more information.
RequestId	string	A Request id.

Comments

Abnormal result codes

Name	Description
E_InvalidSessionId	See ResultCode
E_WrongParameter	See ResultCode

SetLogRequestParameters

SetLogRequestParameters

```
<s:element name="SetLogRequestParameters">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="RequestId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="parameters" type="tns:LogRequestParameters" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

SetLogRequestParameters is the container of information that represents the *SetLogRequestParameters* request.

This method provides the ability to modify parameters of a request on the archived events on the server side.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
RequestId	string	The Identifier of the Log request.
parameters	LogRequestParameters	See CreateLogRequest.

Comments

SetLogRequestParametersResponse

```
      <s:element name="SetLogRequestParametersResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SetLogRequestParametersResult"
type="tns:Result" />
          </s:sequence>
        </s:complexType>
       </s:element>
```

Description

SetLogRequestParametersResponse is the container of information that represents the *SetLogRequestParameters* response.

Parameters

Name	Type	Description
SetLogRequestParametersResult	Result	Required element. See abnormal result codes for more information.

Comments

Abnormal result codes

Name	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode
E_WrongParameter	See ResultCode

QueryLogRequest

Querying log requests can be done for different time periods definitions.
It is possible to request:

- *n* records from the beginning of archived data without passing any timedate to the SV web server.
- *n* records until the end of archived data without passing any timedate to the SV web server.
- *n* records before a given timedate passed as parameter when invoking the service.
- *n* records after a given timedate passed as parameter when invoking the service.
- *n* records between 2 timedates passed as parameters when invoking the service.

Where *n* is a parameter of the request.

This eases the design of a log viewer, in particular the time scrolling management with capabilities like:

- Show first logged events (mode *Begin*).
- Show last logged events (mode *End*).
- Show previous logged events (mode *Before*).
- Show next logged events (mode *After*).
- Show logged events for a given time period (mode *Between*).

Due to XML/SOAP Web Services limitations with regards to optional parameters handling, these 5 requests are available using 5 different methods describe below.

Once a log request is created, you can call any of these 5 methods, in the order you want, and as many time as necessary before closing the request.

QueryLogRequestBegin

```
<s:element name="QueryLogRequestBegin">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="RequestId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

QueryLogRequestBegin is the container of information that represents the *QueryLogRequestBegin* request.

This method provides the ability to get archived events corresponding to a log request.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
RequestId	string	The Identifier of the request.

Comments

For a given *RequestId*, this method returns *n* records from the beginning of archived data without passing any timedate to the SV web server. *n* is the *ElementMaxNumber* passed as parameter when creating the log request.

QueryLogRequestBeginResponse

```
      <s:element name="QueryLogRequestBeginResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="QueryLogRequestBeginResult" type="tns:Result"
/>
            <s:element minOccurs="0" maxOccurs="1" name="replyData" type="tns:ArrayOfLogValue" />
            <s:element minOccurs="1" maxOccurs="1" name="MaxNumberExceeded" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

QueryLogRequestBeginResponse is the container of information that represents the *QuesryLogRequestBegin* response.

Parameters

Name	Type	Description
QueryLogRequestBeginResult	Result	Required element. See abnormal result codes for more information.
replyData	ArrayOfLogValue	The array of data (logged events and properties).
MaxNumberExceeded	boolean	Feature limitation 8: *HistoricalData–Request*: The *MaxNumberExceeded* flag is not implemented on server side. It is always returned *False*.

Comments

Abnormal result codes

Error Code	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode

QueryLogRequestEnd

```
<s:element name="QueryLogRequestEnd">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="RequestId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

QueryLogRequestEnd is the container of information that represents the *QueryLogRequestEnd* request.

This method provides the ability to get archived events corresponding to a log request.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
RequestId	string	The Identifier of the request.

Comments

For a given *RequestId*, this method returns *n* records until the end of archived data without passing any timedate to the SV web server. *n* is the *ElementMaxNumber* passed as parameter when creating the log request.

QueryLogRequestEndResponse

```
<s:element name="QueryLogRequestEndResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="QueryLogRequestEndResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="replyData" type="tns:ArrayOfLogValue" />
      <s:element minOccurs="1" maxOccurs="1" name="MaxNumberExceeded" type="s:boolean" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

QueryLogRequestEndResponse is the container of information that represents the *QuesryLogRequestEnd* response.

Parameters

Name	Type	Description
QueryLogRequestEndResult	Result	Required element. See abnormal result codes for more information.
replyData	ArrayOfLogValue	The array of data (logged events and properties).
MaxNumberExceeded	boolean	Feature limitation MaxNbExceededFlag applies.

Comments

Abnormal result codes

Error Code	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode

QueryLogRequestBefore

```
<s:element name="QueryLogRequestBefore">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="RequestId" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="BeforeTimedate" type="s:dateTime" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

QueryLogRequestBefore is the container of information that represents the *QueryLogRequestBefore* request.

This method provides the ability to get archived events corresponding to a log request.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
RequestId	string	The Identifier of the request.
BeforeTimedate	datetime	The timedate to consider

Comments

For a given *RequestId*, this method returns *n* records archived before a given timedate. *n* is the *ElementMaxNumber* passed as parameter when creating the log request.

QueryLogRequestBeforeResponse

```
      <s:element name="QueryLogRequestBeforeResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="QueryLogRequestBeforeResult" type="tns:Result"
/>
            <s:element minOccurs="0" maxOccurs="1" name="replyData" type="tns:ArrayOfLogValue" />
            <s:element minOccurs="1" maxOccurs="1" name="MaxNumberExceeded" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

QueryLogRequestBeforeResponse is the container of information that represents the *QuesryLogRequestBefore* response.

Parameters

Name	Type	Description
QueryLogRequestBeforeResult	Result	Required element. See abnormal result codes for more information.
replyData	ArrayOfLogValue	The array of data (logged events and properties).
MaxNumberExceeded	boolean	Feature limitation MaxNbExceededFlag applies.

Comments

Abnormal result codes

Error Code	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode

QueryLogRequestAfter

```
<s:element name="QueryLogRequestAfter">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="RequestId" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="AfterTimedate" type="s:dateTime" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

QueryLogRequestAfter is the container of information that represents the *QueryLogRequestAfter* request.

This method provides the ability to get archived events corresponding to a log request.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
RequestId	string	The Identifier of the request.
AfterTimedate	datetime	The timedate to consider

Comments

For a given *RequestId*, this method returns *n* records archived after a given timedate. *n* is the *ElementMaxNumber* passed as parameter when creating the log request.

QueryLogRequestAfterResponse

```
      <s:element name="QueryLogRequestAfterResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="QueryLogRequestAfterResult" type="tns:Result"
/>
            <s:element minOccurs="0" maxOccurs="1" name="replyData" type="tns:ArrayOfLogValue" />
            <s:element minOccurs="1" maxOccurs="1" name="MaxNumberExceeded" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

QueryLogRequestAfterResponse is the container of information that represents the *QuesryLogRequestAfter* response.

Parameters

Name	Type	Description
QueryLogRequestAfterResult	Result	Required element. See abnormal result codes for more information.
replyData	ArrayOfLogValue	The array of data (logged events and properties).
MaxNumberExceeded	boolean	Feature limitation MaxNbExceededFlag applies.

Comments

Abnormal result codes

Error Code	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode

QueryLogRequestBetween

```
<s:element name="QueryLogRequestBetween">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="RequestId" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="StartTime" type="s:dateTime" />
      <s:element minOccurs="1" maxOccurs="1" name="EndTime" type="s:dateTime" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

QueryLogRequestBetween is the container of information that represents the *QueryLogRequestBetween* request.

This method provides the ability to get archived events corresponding to a log request.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
RequestId	string	The Identifier of the request.
StartTime	datetime	The start timedate to consider
EndTime	datetime	The end timedate to consider

Comments

For a given *RequestId*, this method returns *n* records archived between 2 given timedates. *n* is the *ElementMaxNumber* passed as parameter when creating the log request.

QueryLogRequestBetweenResponse

```
      <s:element name="QueryLogRequestBetweenResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="QueryLogRequestBetweenResult"
type="tns:Result" />
            <s:element minOccurs="0" maxOccurs="1" name="replyData" type="tns:ArrayOfLogValue" />
            <s:element minOccurs="1" maxOccurs="1" name="MaxNumberExceeded" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

QueryLogRequestBetweenResponse is the container of information that represents the *QuesryLogRequestBetween* response.

Parameters

Name	Type	Description
QueryLogRequestBetweenResult	Result	Required element. See abnormal result codes for more information.
replyData	ArrayOfLogValue	The array of data (logged events and properties).
MaxNumberExceeded	boolean	Feature limitation MaxNbExceededFlag applies.

Comments

Abnormal result codes

Error Code	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode

CloseLogRequest

CloseLogRequest

```
<s:element name="CloseLogRequest">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="RequestId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

CloseLogRequest is the container of information that represents the *CloseLogRequest* request.

This method provides the ability to close a request to archived events on the server side, so that the server is able to release the request context if the client is no more willing to get those archived events data.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
RequestId	string	The Identifier of the Request.

Comments

This method must be called by a client prior to close a session for every request it has created.

CloseLogRequestResponse

```
<s:element name="CloseLogRequestResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="CloseLogRequestResult" type="tns:Result" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

CloseLogRequestResponse is the container of information that represents the *CloseLogRequest* response.

Parameters

Name	Type	Description
CloseLogRequestResult	Result	Required element. See abnormal result codes for more information.

Comments

Abnormal result codes

Error Code	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode

ExecuteLogRequest

ExecuteLogRequest

```
<s:element name="ExecuteLogRequest">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="parameters" type="tns:LogRequestParameters" />
      <s:element minOccurs="1" maxOccurs="1" name="StartTime" type="s:dateTime" />
      <s:element minOccurs="1" maxOccurs="1" name="EndTime" type="s:dateTime" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

ExecuteLogRequest is the container of information that represents the *ExecuteLogRequest* request.

This method provides the ability to get archived events in a single request.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
parameters	LogRequestParameters	Log parameters for the request
StartTime	datetime	The start timedate to consider
EndTime	datetime	The end timedate to consider

Comments

This method returns *n* records archived between 2 given timedates. *n* is the *ElementMaxNumber* passed as parameter.

ExecuteLogRequestResponse

```
<s:element name="ExecuteLogRequestResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="ExecuteLogRequestResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="replyData" type="tns:ArrayOfLogValue" />
      <s:element minOccurs="1" maxOccurs="1" name="MaxNumberExceeded" type="s:boolean" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

ExecuteLogRequestResponse is the container of information that represents the *ExecuteLogRequest* response.

Parameters

Name	Type	Description
ExecuteLogRequestResult	Result	Required element. See abnormal result codes for more information.
replyDataList	ArrayOfLogValue	The array of data (logged events and properties).
MaxNumberExceeded	boolean	Feature limitation MaxNbExceededFlag applies.

Comments

Abnormal result codes

Error Code	Description
E_InvalidSessionId	See ResultCode
E_WrongParameter	See ResultCode

GetLogLists

GetLogLists

```
<s:element name="GetLogLists">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

GetLogLists is the container of information that represents the *GetLogLists* request.

This method provides the ability to retrieve names of log lists managed by the server.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.

Comments

GetLogListsResponse

```
<s:element name="GetLogListsResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="GetLogListsResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="logLists" type="tns:ArrayOfLogList" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

GetLogListsResponse is the container of information that represents the *GetLogLists* response.

Parameters

Name	Type	Description
GetLogListsResult	Result	Required element. See abnormal result codes for more information.
logLists	ArrayOfLogList	The list of log lists

Comments

Abnormal result codes

Error Code	Description
E_InvalidSessionId	See ResultCode

GetFilters

GetFilters

```
<s:element name="GetFilters">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="projectLanguage" type="tns:SVLanguage" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

GetFilters is the container of information that represents the *GetFilters* request.

This method provides the ability to retrieve the set of predefined filters managed by the server.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
projectLanguage	SVLanguage	SV language for the request. If not set, the project language of the session is used.

Comments

See Predefined filters on server side.

GetFiltersResponse

```
<s:element name="GetFiltersResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="GetFiltersResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="filters" type="tns:ArrayOfFilter" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

GetFiltersResponse is the container of information that represents the *GetFilters* response.

Parameters

Name	Type	Description
GetFiltersResult	Result	Required element. See abnormal result codes for more information.
filters	ArrayOfFilter	An array containing the filters configured on server side for logged events.

Comments

Abnormal result codes

Name	Description
E_InvalidSessionId	See ResultCode

CreateTrendRequest

CreateTrendRequest

```
      <s:element name="CreateTrendRequest">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="parameters" type="tns:TrendRequestParameters"
/>
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

CreateTrendRequest is the container of information that represents the *CreateTrendRequest* request.

This method provides the ability to create a request on the archived trends on the server side, so that the server is able to have a persistent request the client may use to retrieve data for different time periods without passing a complete parameter set.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
parameters	TrendRequestParameters	Trend parameters for the request.

Comments

CreateTrendRequestResponse

```
<s:element name="CreateTrendRequestResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="CreateTrendRequestResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="RequestId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

CreateTrendRequestResponse is the container of information that represents the *CreateTrendRequest* response.

Parameters

Name	Type	Description
CreateTrendRequestResult	Result	Required element. See abnormal result codes for more information.
RequestId	string	A Request id

Comments

Abnormal result codes

Error Code	Description
E_InvalidSessionId	See ResultCode
E_WrongParameter	See ResultCode

SetTrendRequestParameters

SetTrendRequestParameters

```
      <s:element name="SetTrendRequestParameters">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="RequestId" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="parameters" type="tns:TrendRequestParameters"
/>
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

SetTrendRequestParameters is the container of information that represents the *SetTrendRequestParameters* request.

This method provides the ability to modify parameters of a request on the archived trend on the server side.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
RequestId	string	The Identifier of the Trend request.
parameters	TrendRequestParameters	See CreateTrendRequest.

Comments

SetTrendRequestParametersResponse

```
      <s:element name="SetTrendRequestParametersResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SetTrendRequestParametersResult"
type="tns:Result" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

SetTrendRequestParametersResponse is the container of information that represents the *SetTrendRequestParameters* response.

Parameters

Name	Type	Description
SetTrendRequestParametersResult	Result	Required element. See abnormal result codes for more information.

Comments

Abnormal result codes

Name	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode
E_WrongParameter	See ResultCode

QueryTrendRequest

QueryTrendRequest

```
<s:element name="QueryTrendRequest">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="RequestId" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="StartTime" type="s:dateTime" />
      <s:element minOccurs="1" maxOccurs="1" name="EndTime" type="s:dateTime" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

QueryTrendRequest is the container of information that represents the *QueryTrendRequest* request.

This method provides the ability to get archived data corresponding to a request to archived trend.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
RequestId	string	The Identifier of the request.
StartTime	datetime	The start timedate to consider
EndTime	datetime	The end timedate to consider

Comments

For a given *RequestId*, this method returns *n* records archived between 2 given timedates. *n* is the *ElementMaxNumber* passed as parameter when creating the trend request.

QueryTrendRequestResponse

```
<s:element name="QueryTrendRequestResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="QueryTrendRequestResult" type="tns:Result" />
      <s:element minOccurs="0" maxOccurs="1" name="replyData" type="tns:ArrayOfTrendValue" />
      <s:element minOccurs="1" maxOccurs="1" name="MaxNumberExceeded" type="s:boolean" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

QueryTrendRequestResponse is the container of information that represents the *QueryTrendRequest* response.

Parameters

Name	Type	Description
QueryTrendRequestResult	Result	Required element. See abnormal result codes for more information.
replyData	ArrayOfTrendValue	The array of data (archived trend data).
MaxNumberExceeded	boolean	If false, it indicates that all data matching the request were returned. If true, more data matches the request, but not all of them could be returned in a single response. The request should be re-iterated for the interval of time for which no data was returned.

Comments

Abnormal result codes

Error Code	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode

CloseTrendRequest

CloseTrendRequest

```
<s:element name="CloseTrendRequest">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="RequestId" type="s:string" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

CloseTrendRequest is the container of information that represents the *CloseTrendRequest* request.

This method provides the ability to delete a request to trend data on the server side, so that the server is able to release the context if the client is no more willing to get those trend data.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
RequestId	string	The Identifier of the trend request.

Comments

This method must be called by a client prior to close a session for every request it has created.

CloseTrendRequestResponse

```
<s:element name="CloseTrendRequestResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="CloseTrendRequestResult" type="tns:Result" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Description

CloseTrendRequestResponse is the container of information that represents the *CloseTrendRequest* response.

Parameters

Name	Type	Description
CloseTrendRequestResult	Result	Required element. See abnormal result codes for more information.

Comments

Abnormal result codes

Error Code	Description
E_InvalidSessionId	See ResultCode
E_InvalidId	See ResultCode

ExecuteTrendRequest

ExecuteTrendRequest

```
      <s:element name="ExecuteTrendRequest">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="parameters" type="tns:TrendRequestParameters"
/>
            <s:element minOccurs="1" maxOccurs="1" name="StartTime" type="s:dateTime" />
            <s:element minOccurs="1" maxOccurs="1" name="EndTime" type="s:dateTime" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

ExecuteTrendRequest is the container of information that represents the *ExecuteTrendRequest* request.

This method provides the ability to get archived trend in a single request.

Parameters

Name	Type	Description
SessionId	string	A valid *SessionId*.
parameters	TrendRequestParameters	Trend parameters for the request
StartTime	datetime	The start timedate to consider
EndTime	datetime	The end timedate to consider

Comments

This method returns *n* records archived between 2 given timedates. *n* is the *ElementMaxNumber* passed as parameter.

ExecuteTrendRequestResponse

```
      <s:element name="ExecuteTrendRequestResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="ExecuteTrendRequestResult" type="tns:Result"
/>
            <s:element minOccurs="0" maxOccurs="1" name="replyData" type="tns:ArrayOfTrendValue" />
            <s:element minOccurs="1" maxOccurs="1" name="MaxNumberExceeded" type="s:boolean" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

Description

ExecuteTrendRequestResponse is the container of information that represents the *ExecuteTrendRequest* response.

Parameters

Name	Type	Description
ExecuteTrendRequestResult	Result	Required element. See abnormal result codes for more information.
replyData	ArrayOfTrendValue	The array of data (archived trend data).
MaxNumberExceeded	boolean	If false, it indicates that all data matching the request were returned. If true, more data matches the request, but not all of them could be returned in a single response. The request should be re-iterated for the interval of time for which no data was returned.

Comments

Abnormal result codes

Error Code	Description
E_InvalidSessionId	See ResultCode
E_WrongParameter	See ResultCode

Annex 1: Feature limitations

Feature limitation 1: *ResultCode*: E_*MaxExceeded* is not implemented. No method returns this error code. 19
Feature limitation 2 : *RealTimeAlarm–AlarmSupportedEventCodes*: The alarm state *AlarmTakenIntoAccount* is not implemented on the *Web Services Toolkit* server side. This event code must not be added to filter definition when using the *Subscribe*, *SetSubscriptionParameters* or the *Read* methods. 60
Feature limitation 3: *RealTimeAlarm*: The *MaxNumberExceeded* flag is not implemented on server side. It is always returned *False*. 69
Feature limitation 4 : *RealTimeAlarm-ExecuteUserAction*: The user actions *SetTakenIntoAccount* and *ResetTakenIntoAccount* are not implemented yet. Invoking this method for one of these user action code returns *S_Ok* whilst no action is performed on server side. 76
Feature limitation 5 : *HistoricalData*: GPConf.dat filters are not implemented for HDS archive units. For HDS archive units, the *filters* item must be left empty. 79
Feature limitation 6 : *HistoricalData-TrendRequestParameters*: The *IncludeEndBound* option is only implemented for HDS archive units. Set it to the value *False* for Proprietary archive units (not implemented). 86
Feature limitation 7 : *HistoricalData- TrendRequestParameters*: Access to additional properties is not supported for trend requests. The *properties* item must not be used (null). 86
Feature limitation 8: *HistoricalData–Request*: The *MaxNumberExceeded* flag is not implemented on server side. It is always returned *False*. 94

Annex 2: WebVue Java applet - Advanced use

Introduction to WebVue

See SV documentation for more information on WebVue.

WebVue is a client-server application designed to display SV mimics on a Web browser.
WebVue has 2 components:

- A light client (known as the WebVue Client): An applet that runs under the Java Virtual Machine in a Web browser.
- A server (known as the WebVue Server) that is an SV component.

The WebVue Client and WebVue Server communicate using a dedicated XML/SOAP Web Service.

WebVue uses 3 virtual directories:

- *Classes* – Used by the Web browser to load the Java applet.
- *WebPropertyServer* – Used by the WebVue applet to exchange data with its SV server (Uri of the dedicated XML/SOAP Web Service).
- <*Project name*> - Used by the WebVue applet to access configuration of the project. <*Project name*> is replaced by the actual name of the project.

Using WebVue does not require any special knowledge. Once WebVue is configured on server side, users can connect to it:

1. By using the URL of the virtual directory <*Project name*>.
2. An HTML page will appear containing nothing more than the applet instanciation. The HTML page is physically located in the *Web* sub-directory of the SV project (*index.html*)
3. The applet is loaded.
4. A login and a password are asked (SV valid account).
5. If the login is successful, the initial mimic of the user is displayed. To do so, the WebVue applet initiates a session with the SV server.
6. The user can use mimics according to the application design and the rights he is granted.
7. Closing the Web browser closes the communication channel between the applet and the server.

WebVue and the OpenCustomSession method

One of the assets of WebVue is that it completely relies on the user profile configuration. Profiles define:

- The language in which mimics are displayed.
- The initial mimic after login.
- …

But this implies that the WebVue login is not optional.

In some contexts, for example the integration of the WebVue applet in third-party Web portals, the applet behaviour must not only rely on the login and the password given by the user in front of the Web browser, but also in the browsing context in the Web portal.
To allow smooth integration of the WebVue applet in third party portals, the *Web Service Toolkit* offers a method called *OpenCustomSession* that allows opening a session for WebVue and overriding the configuration of the user profile. Openning such a session and passing the Session Id to the WebVue applet at instanciation time permits to:

- Ignore and override any piece of configuration of the user profile, in particular the language used.
- Achieve an automatic login to the WebVue server, because, as the WebVue applet does not need to open a session itself, it does not need to ask any login information to the user (the login dialog box does not open in this case) but uses an already existing session.
- Instantiate the WebVue applet and display an arbitrary mimic of the SV application.

Using OpenCustomSession

To instantiate the WebVue applet in an arbitrary user context:

1. Use the *OpenCustomSession* method to get a *SessionId* with the user context you want. The user context is defined on server side on the base of the parameters passed when calling *OpenCustomSession*. The *Username* and *Password* must correspond to a valid SV account with WebVue access granted, but all other item of the user context are defined by the *SessionParameters* element: *InitialWindow*, *InitialWindowBranch*, *projectLanguage*, *AlarmFilter*, *LogFilter* and *BeepOnNewAlarm*.
2. Instantiate the WebVue applet in your custom Web page (see below a sample).
3. Add the *SESSIONID* parameter to the piece of HTML code. The value of the *SESSIONID* parameter is the Session Id obtained by the call to *OpenCustomSession*.

Doing so allow the user to have the WebVue applet loading and initializing:

- Without explicite login.
- With a first mimic you have chosen.
- With a language context you have chosen.
- …

HTML sample:

Replace *SESSION_ID* by the actual value of Session Id returned by the call to *OpenCustomSession*.

```
var attributes = {
        code: 'svfc/launcher/WebVueClient.class',
        archive: 'AiWebVue.jar',
        codebase: '/classes',
        width: '100%',
        height: '100%',
        align: 'baseline',
        sessionid: 'SESSION_ID'
};
```

The WebVue applet automatically closes the session it uses when it is freed. You do not need to call *CloseSession* as soon as the WebVue applet as been instantiated.

Overriding the user profile

Using the *OpenCustomSession* prior to instantiating the WebVue applet allows you to override any piece of configuration of the user profile (WebVue settings only).

It is useful if you wish to use a generic SV account whatever the user using the third-party Web Portal, but want to keep the freedom of customizing the language or the initial mimic.
In particular, you can use this feature to use the WebVue applet as a mimic viewer.

Automatic login – Single Sign On for WebVue

You can use this feature to implement a Single Sign On procedure for WebVue.
According to the user context in your third-party Web portal, open a WebVue session in background and instantiate the applet in the context you want without requiring an explicit login for WebVue.

Using the WebVue applet in secured environments

SSL v3

The WebVue applet supports SSL v2 using X509 certificates, but it does not support SSL v3.

Assuming that the WebVue server and your Web portal server are on the same secured network, you can:

- Implement SSL v3 between the web clients and your Web portal,
- Let your web portal manage SSL client certificates.
- Deploy SSL v2 for communications between your Web portal and the SV Web server by customizing the IIS configuration of virtual directories used by the SV Web Server.
- Use the *OpenCustomSession* on https (SSL v2) to obtain Session Ids by using the SV account you want according to the SSL client certificate of the requester.

It allows you to have web clients' identification and authentication using SSL client certificates, and let the WebVue applet communicates with the WebVue server over an SSL v2 channel:

- Identification and authentication of clients from the public network is SSL v3 based,
- Web clients are identified with their own SSL certificates,
- The communication channel between the SV Web server and your Web Portal on a secured private network is based on SSL v2.

Unusual Java applet hosting

For security reason, it happens that the WebVue applet must be hosted on a server different from the server running SV.

It is possible to do so by adding a parameter called *hosturl* to the WebVue applet instantiation. The value of this parameter must be the http address of the server hosting the following virtual directories:

- WebPropertyServer
- <Project name>.

It allows the WebVue applet to be loaded from one host and to connect to another host for the runtime communication channel with its WebVue server.
A possible value for the parameter hosturl is http://*ServerIPAddress* if the virtual directories mentioned are available on this server.

Note:

In such architecture, the SV virtual directory named *Classes* is no more necessary.
Be careful to the fact that the WebVue applet must be updated on its host each time a change of version occurs for the SV Web Server.

The default Java policy does not allow a Java applet to connect to an IP address if it is not the one from where it has been loaded. To change this, the following item must be added to the Java policy on web client machines:

```
grant
{permission java.net.SocketPermission "ServerIPAddress:ServerPort-", "accept,connect,listen,resolve";}
```

Where *ServerIpAddress* is the address of the SV Web Server, and *ServerPort* is the port to use (usually 80 or 443).

Another option is to use a digitally signed WebVue applet (see below).

Signed Java applet

It happens that the WebVue applet needs to be digitally signed.

To do so, you can repackage the WebVue jar files using you own digital certificate.
Necessary Jar files can be found in the SV directory *\BIN\WebServerExtensions\WebVue\Classes*.

Do not forget to change the WebVue applet instantiation by referencing your own package in the parameter named *archive*.

Note:

Be careful to the fact that your package must be updated each time a change of version occurs for the SV Web Server.

Annex 3: Properties management

Please read this annex carefully if you wish to access data properties

It contains important information concerning properties handling by the different services of the *Web Services Toolkit*

Properties are accessible for all types of data:

- Real time variable: *Read*, *GetProperties* and *Subscribe*.
- Real time alarm: *Read*, *Subscribe*.
- Logged events: All types of query.
- Trend data: All types of query.

Each service defines its own enumeration of available properties:

- VariableProperty for the *RealTimeData* service.
- AlarmProperty for the *RealTimeAlarm* service.
- LogProperty and TrendProperty for the *HistoricalData* service.

The 4 enumerations are hardly the same, but wisely using these enumerations require the user to be aware of some major differences in the data handling by the SV Web server. Differences fall in 3 categories:

- **Use of some properties is reserved for future in the context of one or more service:**
  At best they are not in the corresponding enumeration. Nevertheless they sometimes are.
- **Some properties are always returned empty by one or more service:**
  This is due to SV Web Server limitations preventing web client from getting the correct information. Limitations in this category will be cleared in future versions.
  Such properties can be requested, but empty strings are returned whatever the actual value on SV side.
- **Some properties are not managed at all and must not be requested:**
  This is due to SV internal limitations, preventing the SV Web Server from getting the information. Limitations in this category may be cleared in future versions.
  Such properties must not be requested. Including one of these properties may lead to incorrect processing on SV server side.

Properties in categories 2 and 3 are in enumerations to avoid breaking compatibility when the limitations will be cleared.

For each enumeration, the table below explains the SV Web server behaviour properties by properties:

- **OK** - Can be used.
- **Empty** - Always returned empty (temporary behaviour until correction).
- **Reserved** - Not managed, must not be requested.

Green cell background indicates the property is in the corresponding enumeration.
No cell background indicates the property is not in the corresponding enumeration.

Name	VariableProperty	AlarmProperty	LogProperty	TrendProperty	Description
VariableName	OK	OK	OK	Reserved	Variable name (#t).
Description	OK	OK	OK	Reserved	Variable title (#T).
TimestampType	OK	OK	OK	Reserved	PLC timestamp or SV timestamp (#x).
AssocLabel	Empty	OK	OK	Reserved	The associated Label (#E).
StandardLabel		OK	OK	Reserved	Default associated label (#L).
UserName		OK	OK	Reserved	Operator Name (#N).
Station		OK	OK	Reserved	Station Number (#S).
AlarmLevel		OK	OK	Reserved	Alarm Level (#A).
Informations		OK	OK	Reserved	Threshold or value (#C)
LogInformations		Reserved	OK	Reserved	Comment (#c)
TextAttr01 to TextAttr16	OK	OK	OK	Reserved	Text attributes 1 to 16. Text attribute 1: Domain (#d or #@A1). Text attribute 2: Nature (#n or #@A2). Other Text attributes (#@A3 to #@A16).
DeferredTextAttr03 to DeferredTextAttr16	OK	OK	OK	Reserved	Deferred Text attributes 3 to 16 (#@*A3 to #@*A16).
BinAttr	OK	OK	OK	Reserved	Binary attribute (#B).
MinValue	OK (1)				The Minimum Value at the time of request. It can be a configured value, a value modified by script, or the value of the Minimum variable in case it is defined by an indirection on another variable.
MaxValue	OK (1)				The Maximum Value at the time of request. It can be a configured value, a value modified by script, or the value of the Maximum variable in case it is defined by an indirection on another variable.
MinControlValue	OK (1)				The Minimum Control Value at the time of request. It can be a configured value, a value modified by script, or the value of the Minimum Control variable in case it is defined by an indirection on another variable.
MaxControlValue	OK (1)				The Maximum Control Value at the time of request. It can be a configured value, a value modified by script, or the value of the Maximum Control variable in case it is defined by an indirection on another variable.
MinVariableName	OK (1)				The name of the variable in case it is defined by an indirection on another variable.

Name	VariableProperty	AlarmProperty	LogProperty	TrendProperty	Description
MaxVariableName	OK (1)				The name of the variable in case it is defined by an indirection on another variable.
MinControlVariableName	OK (1)				The name of the variable in case it is defined by an indirection on another variable.
MaxControlVariableName	OK (1)				The name of the variable in case it is defined by an indirection on another variable.
Unit	OK				Engineering unit.
Format	OK				Default output format.
Anim0AssocLabel	OK				Associated label base language.
Anim1AssocLabel	OK				Associated label alternative language.
Cmd0AssocLabel	OK				Associated label base language.
Cmd1AssocLabel	OK				Associated label alternative language.
Log0AssocLabel	OK				Associated label base language.
Log1AssocLabel	OK				Associated label alternative language.
FormattedValue	OK				Reserved.

(1) The access to bound values and bound variable names has been validated for the GetProperties method only.