



EMC® Watch4net®

Version 6.4u2

Frontend Web Service Developer Tutorial

P/N: pending

Rev : 50608

EMC²

Copyright ©2008 - 2014 EMC Corporation. All rights reserved. Published in the USA.

Published October, 2014

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

The information in this publication is provided as is. EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose. Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

EMC², EMC, and the EMC logo are registered trademarks or trademarks of EMC Corporation in the United States and other countries. All other trademarks used herein are the property of their respective owners.

For the most up-to-date regulatory document for your product line, go to EMC Online Support (<https://support.emc.com>).

Contents

Preface	iii
Purpose	iii
Intended Audience	iii
Prerequisites and Conventions	iii
Document Organization	iv
Where to get help	v
1 APG Web Service Introduction	1
1 DatabaseAccessor Service	2
2 ReportManager and ReportManagerInline Services	2
3 Quick start tips	2
3.1 Using the provided java-cli	3
2 Java Tutorial	5
1 JAX-WS Introduction	5
1.1 How it works ?	5
1.2 Step-by-Step client application checklist	6
2 Java Sample Application	6
2.1 Overview	6
2.2 Initial steps	7
2.3 Combo Boxes	7
2.4 Device Selection Combo Box	8
2.5 Part Selection Combo Box	9
2.6 The JGraph Component	9
2.7 Putting all together	11
3 PHP Tutorial	13
1 Creating an APG graph with PHP	13
1.1 Overview	13
1.2 Report Generation	14

2 Simple Report Page	15
2.1 Creation of the filter	15
2.2 Retrieving device name(s)	16
2.3 Putting all together	17
A DatabaseAccessor Definition	19
1 Performance Data Access	19
1.1 getObjectData	19
1.2 getAggregatedData	21
2 Topology Data Access	23
2.1 getObjectProperties	23
2.2 getDistinctPropertyValues	25
2.3 getDistinctPropertyRecords	26
3 Topology Base Object Access	28
3.1 getObjectFilters	28
3.2 getObjectCount	29
3.3 getStaticObjectFilters	30
3.4 getStaticObjectCount	31
4 Schema Metadata Access	32
4.1 getAvailableAccessors	32
4.2 getAvailableAggregatePeriods	33
4.3 getAvailableProperties	34
4.4 getSuggestions	35
5 WSDL Service Definition	36
5.1 DatabaseAccessor.wsdl	37
5.2 DatabaseAccessor.xsd	44
B ReportManager Definition	55
1 Report Generation Interface	55
1.1 getReport	55
2 WSDL Service Definition	58
2.1 ReportManager.wsdl	58
2.2 ReportManager.xsd	60
2.3 XmlTree1.xsd	62
2.4 XmlReport1.xsd	105
2.5 xmlmime.xsd	113

Preface

Purpose

This document will assist you in the creation of client applications which connect to the *APG* Web Service. Through simple examples, you will see how to create several kind of applications which remotely use the *APG* Reporting Engine.

Intended Audience

This guide is mainly targeted to developers who want to create third party applications based on the *APG* Reporting Engine. It assumes that you already have a good knowledge of the targeted programming language (*Java*, *PHP*, ...) and that you are familiar with the concepts of Web Services and Remote Procedure Calls. Furthermore, a good understanding of *APG Frontend* reports edition is required (see *APG Frontend* documentation).

Prerequisites and Conventions

All the path mentioned in this document are supposed to be relative to your *APG* installation. It assumes that the *APG* Web Service application has been deployed and is accessible through the URL <http://localhost:58080/APG-WS>. We also assume that we have a single user is configured which username is *admin* and its password is *changeme*. This user is supposed to have unlimited rights on

the application.

Please note that the *XML* message samples in this document are not intended to be syntactically correct (e.g. namespaces were omitted).

Document Organization

1. <i>APG</i> Web Service Introduction	This section briefly explains what the <i>APG</i> Web Service offers.
2. <i>Java</i> Tutorial	This section guides you through a step-by-step example of how to create a simple <i>Java</i> client.
3. <i>PHP</i> Tutorial	This section explains how to create some simple report pages using <i>PHP</i> .

Where to get help

EMC support, product, and licensing information can be obtained as follows:

PRODUCT INFORMATION - For documentation, release notes, software updates, or information about EMC products, go to EMC Online Support at:

<https://support.emc.com>

TECHNICAL SUPPORT - Go to EMC Online Support and click Service Center. You will see several options for contacting EMC Technical Support. Note that to open a service request, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or with questions about your account.

APG Web Service Introduction

APG Web Service leverages Enterprise Standards like *SOAP* and *WSDL* to offer its functionality. It is composed of two main services:

- the *DatabaseAccessor* service provide low level access to the *APG* databases.
- the *ReportManager* and *ReportManagerInline* services provides access to the reporting engine. The two services offer the same functionality, the only difference is the format of their messages.

Note: Please note that the default *APG* Web Service configuration only allows access from *localhost* and the authentication is enabled. To remove the access restriction, comment the *RemoteAddrValve* element in *APG-WS.xml* (located in *<Tomcat>/conf/Catalina/localhost/*). To disable the authentication, edit *web.xml* (located in *<Tomcat>/webapps/APG-WS/WEB-INF*) and comment the corresponding section.

1 *DatabaseAccessor Service*

This service allow third party applications to access the data at the lowest level available in *APG*. Data returned by this service is not processed or formatted in any way, it directly comes from your *APG* databases. For a complete listing of available functions, see appendix [A](#) on page [19](#).

This service definition is accessible through the url <http://localhost:58080/APG-WS/wsapi/db?wsdl>.

2 *ReportManager and ReportManagerInline Services*

These two services can be used to generate reports in a neutral and versatile *XML* format. The report generation and rendering process can be customized using properties and the actual report generation works exactly the same way it does in the *APG* Web Portal. The input *XML* tree is in the same format, and then, exports from the Web Portal can be used directly as a parameter of the service method. For more information, see appendix [B](#) on page [55](#).

The *ReportManager* and *ReportManagerInline* services has exactly the same definition, however, the *ReportManager* service sends its response using *MTOM* encoding (response is a multi part document and images are sent as attachments) while the *ReportManagerInline* service use conventional *XML* response with *Base64* encoded binary data.

The service definitions are accessible through the urls <http://localhost:58080/APG-WS/wsapi/report?wsdl> and http://localhost:58080/APG-WS/wsapi/report_inline?wsdl.

3 *Quick start tips*

A good way to start using *APG* Web Service is to firstly get an eye on *APG* reports in *APG Frontend*. *XML* exports provide all necessary informations to build a report in *APG* Web Service. A good understanding of the report *XML* tree (*xmlTree1.xsd*) is required to use the *ReportManagers* web services.

Note: *xmlTree1.xsd* contains necessary parameters and documentation to get reports through *APG* Web Service.

Steps to setup a report through *APG* Web Service Report manager:

1. Find or create in *APG Frontend* a report wanted through *APG* Web Service

2. Go in *Edit Mode* and get the *Report Definition* in *Exports* menu.
3. Open the *XML* file to see the report parameters
4. Use the *XML* schema (*xmlTree1.xsd*) to learn about other values and parameters (see [2.3](#))
5. Test using a web service browser (like soapUI <http://www.soapui.org/>). *The node element from the XML export can be directly used as the node parameter in the getReport operation.*
6. Implement it using Java, PHP or any other language.

Note: If the wanted report uses filter in parent nodes or uses level-up from complex expansion going over the current node, you will need to extract the report on a parent node. It is possible to select which report we want by using the `apg.ws.node.id` property.

The `apg.ws.node.id` property can be used to select a report in a XML tree. By default, if no id is given, the apgws will process the root report.

```
[...]
<properties>
<property key="apg.ws.node.id">[nodeID]</property>
</properties>
<node [...]
```

3.1 Using the provided java-cli

A simple client is available to access the *DatabaseAccessor* service. This client provides access to every operation exposed by the *DatabaseAccessor* service. The tool is located in `samples/java-cli`. For more information, please refer to the `README.txt` file.

Here is an Example:

We want to expand on all devices and parts. We use the *GetDistinctPropertyRecords* call with two properties.

```
java -jar apg-remote-db-cli.jar http://localhost:58080/APG-WS/wsapi/db?wsdl \
'GetDistinctPropertyRecords( Property => "device", Property => "part" )'
```

The result will look like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<getDistinctPropertyRecordsResponse xmlns="http://www.watch4net.com/APG/Remote/
DatabaseAccessorService">
  <record>
    <value>device1</value>
    <value>part1</value>
  </record>
  <record>
    <value>device1</value>
    <value>part2</value>
  </record>
  <record>
    <value>device2</value>
    <value>part2</value>
  </record>
</getDistinctPropertyRecordsResponse>
```

2

***Java* Tutorial**

1 JAX-WS Introduction

JAX-WS is the *Java* Web Service framework *APG* is built on and which will be used in this example. It greatly simplifies Web Service development and hides most of the complexity of the actual implementation. This section will introduce you to this framework and show you how to leverage it's simplicity to create an *APG* Web Service client easily.

1.1 How it works ?

JAX-WS is primarily intended to avoid the manipulation of complex *XML* structure by the client applications. That is, *JAX-WS* will automatically create *Java* classes which will be used to generate and parse the actual messages. From the client application point of view, invoking a Web Service method is just like invoking a traditional *Java* method.

1.2 Step-by-Step client application checklist

Here is a short step-by-step list of the task you will have to do when you are developing a client application using *JAX-WS*:

1. Make sure you have the *JAX-WS 2.1 Reference Implementation*. It can be downloaded at <http://jax-ws.java.net/>.
2. Locate the *WSDL* service definition location (e.g. <http://localhost:58080/APG-WS/wsapi/db?wsdl>).
3. Import the service definition using the `wsimport` tool provided by *JAX-WS* (now included in the latest *Sun Java Development Kit*). A typical invocation would be:

```
<JAX-WS>/bin/wsimport http://localhost:58080/APG-WS/wsapi/db?wsdl -d bin -s src
```

4. Instantiate the service class (e.g. `DatabaseAccessorService`) and create a service port (e.g. `DatabaseAccessor`) using the right method (e.g. `getDatabaseAccessorPort`).
5. Invoke the remote methods from the service port as traditional *Java* methods.

2 Java Sample Application

This section will guide you through a step-by-step creation of an *APG* Web Service *Java* client application. The files related to this sample are located in `samples/java-graph`.

2.1 Overview

Our client application is a simple grapher, which input is a device name and a part name and that will graph the associated metrics over the last day. The figure 1 on page 7 illustrates the expected Look & Feel of the application.

To assist the user in device and part selection, the application will suggest values based on partial input. For example, device values will be suggested using the beginning of the name and the complete part list will be updated whenever the user selects a new device.

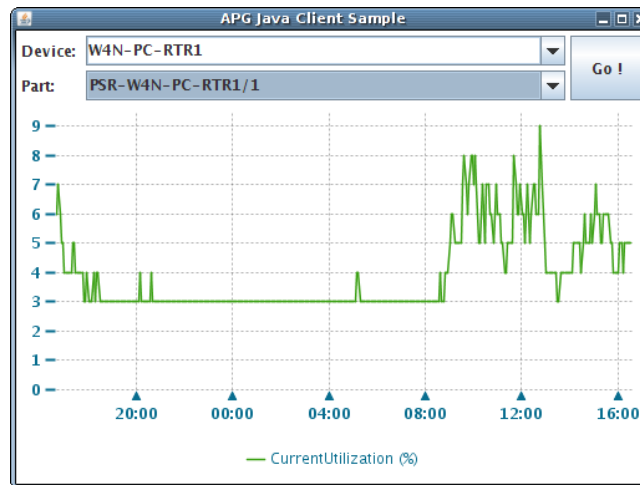


Figure 1: APG Web Service Java Sample Application

2.2 Initial steps

As explained in section 1.2, we initially imports our two service definitions using `wsimport`. Then we can create instances of `DatabaseAccessorService` and `ReportManagerService` to respectively access the *DatabaseAccessor* and *ReportManager* services.

```
<JAX-WS>/bin/wsimport http://localhost:58080/APG-WS/wsapi/db?wsdl \  
-d bin -s src -p com.watch4net.apg.v2.remote.sample.jaxws.db  
  
<JAX-WS>/bin/wsimport http://localhost:58080/APG-WS/wsapi/report?wsdl \  
-d bin -s src -p com.watch4net.apg.v2.remote.sample.jaxws.report
```

2.3 Combo Boxes

To provide suggestions, we will create our own `ComboBoxModel` implementations. In the *Swing* terminology, the `ComboBoxModel` is the class which is responsible for managing the items of a `JComboBox`

component.

To make implementations easier, we create an abstract `ComboBoxModel` which monitors user input and check whether suggested values should be refreshed or not (basically when user input changes, we should refresh suggestions).

The user interaction is reflected by a simple object (called `Context`) which has only three optional properties:

- a property name to indicate what property we should retrieve values from (e.g. `device` for device names and `part` for part names)
- a filter expression to restrict the suggested values (for example, we should only show the parts of the selected device)
- a pattern, which reflect a partial input (for example, the user already typed the beginning of a device name)

Then, subclasses only have to implement an `update(Context ctx)` method which should recreate the suggestion list based on the supplied context.

2.4 Device Selection Combo Box

The device selection combo box has to provide device name suggestions based on a partial name. It's `update()` implementation is:

```
// The port object has been created with:
// port = new DatabaseAccessorService().getDatabaseAccessorPort();
protected void update(Context ctx) throws Exception {
    // retrieve 10 device name suggestions based on partial input
    // we accept * and ? wildcards
    List<Suggestion> vals = port.getSuggestions(ctx.getFilter(), ctx
        .getPattern().replace('*', '%').replace('?', '_')
        + "%", Collections.singletonList(ctx.getProperty()), 10);
    // update item list with the suggested values
    removeAllElements();
    for (Suggestion val : vals) {
        addElement(val.getValue());
    }
}
```

2.5 Part Selection Combo Box

The part selection combo box has to use the previously entered device name to retrieve the complete part list. Its `update()` implementation is:

```
// The port object has been created with:
// port = new DatabaseAccessorService().getDatabaseAccessorPort();
protected void update(Context ctx) throws Exception {
    // retrieve all the part names using the selected device
    List<PropertyRecord> vals = port.getDistinctPropertyRecords(ctx
        .getFilter(), Collections.singletonList(ctx.getProperty()));
    // update item list with the returned values
    removeAllElements();
    for (PropertyRecord propertyRecord : vals) {
        addElement(propertyRecord.getValue().get(0));
    }
}
```

2.6 The JGraph Component

Once device and part have been selected, we want to show the graph of the selected metrics. To do that, we create a new specific *Swing* component called `JGraph`. A `JGraph` takes a filter expression as input and shows all the selected metrics in a graph image. The `JGraph` component will have to:

- create an *XML* Tree template to generate a graph report (as a user would do in the *APG* Web Portal)
- invoke the *ReportManager* service to generate the report
- retrieve the graph image from the generated report
- handle possible report generation errors (user restrictions, ...)
- update its display with the new image

Here is the `JGraph` method which create the request and retrieve the graph from the service:

```
// The port object has been created with:
// port = new new ReportManagerService().getReportManagerPort(new MTOMFeature());
public void updateGraph(String filterExpr) throws Exception {
    Dimension d = getSize();

    // setup some properties
```

```
ReportProperty p;
ReportProperties properties = new ReportProperties();

p = new ReportProperty();
p.setKey(ReportPropertyKey.APG_WS_REPORT_GRAPH_WITH_IMAGE);
p.setValue("true");
properties.getProperty().add(p);

p = new ReportProperty();
p.setKey(ReportPropertyKey.APG_WS_REPORT_GRAPH_WIDTH);
p.setValue(String.valueOf(d.width));
properties.getProperty().add(p);

p = new ReportProperty();
p.setKey(ReportPropertyKey.APG_WS_REPORT_GRAPH_HEIGHT);
p.setValue(String.valueOf(d.height));
properties.getProperty().add(p);

p = new ReportProperty();
p.setKey(ReportPropertyKey.APG_WS_REPORT_GRAPH_WITH_LEGEND_IMAGE);
p.setValue("true");
properties.getProperty().add(p);

// create a simple tree
RealNode root = new RealNode();

RuntimePreferences runtimePrefs = new RuntimePreferences();
// 11h for last 1 hour (can also be p for previous or c for current)
// time units are defined in xmlTree1.xsd
// It is also possible to give a range like "r:1314725066:1314825066"
//   where the numbers are unix timestamps.
runtimePrefs.setTimeRangeExpression("d:11h");
root.getProperty().add(runtimePrefs);

ReportPreferences prefs = new ReportPreferences();
// NRX for simple graph (see xmlTree1.xsd for complete list
prefs.setDefaultMode(DefaultReportMode.NRX.value());
// Average aggregation.
prefs.setPreferredAggregate(AggregationFunction.AVERAGE);
// period of 10 minutes.
prefs.setPreferredPeriod(3600);
root.getProperty().add(prefs);
```

```
NodeFilter filter = new NodeFilter();
filter.setFilterExpression(filterExpr);
root.getProperty().add(filter);

// call the service
Holder<GraphElement> hg = new Holder<GraphElement>();
Holder<ErrorElement> he = new Holder<ErrorElement>();
port.getReport(properties, root, null, hg, he, null, null);

if (he.value != null) {
    // the report generated an error !
    throw new Exception(he.value.getMessage());
} else if (hg.value != null) {
    // we got the graph
    this.image = (BufferedImage) hg.value.getGraph();
    repaint();
} else {
    // wow, this is pretty unexpected...
    throw new Exception("There's no graph in the response.");
}
}
```

2.7 Putting all together

Now that we have all our components, we can build our application main panel. The main frame is called `ApgClientFrame`. This class simply link all components together. More specifically, it provides `Context` instances to reflect user input. For example, here is the device selection combo box context:

```
final Context deviceChoiceCtx = new Context() {

    public String getFilter() {
        // we don't restrict device selection in any way
        return null;
    }

    public String getPattern() {
        // we retrieve partial user input, replace possible wildcards
        // and automatically add a wildcard at the end (we assume that
        // the user entered the beginning of the device name)
        Object i = deviceChooser.getEditor().getItem();
```

```
        return i == null ? null : String.valueOf(i).replace('*', '%')
            .replace('?', '_') + '%';
    }

    public String getProperty() {
        // the device names are stored in the "device" property
        return "device";
    }
};
```

The part selection combo box context is slightly different:

```
final Context partChoiceCtx = new Context() {

    public String getFilter() {
        // returns the following filter:
        // device==<device name entered by the user>
        return Utils.makeFilter(deviceChoiceCtx.getProperty(), String
            .valueOf(deviceChooser.getSelectedItem()));
    }

    public String getPattern() {
        // we don't have partial input for the part selection
        return null;
    }

    public String getProperty() {
        // part names are stored in the part property
        return "part";
    }
};
```

3

PHP Tutorial

PHP comes with a built-in *SOAP* library since version 5.0.1. This example requires *PHP* built with *SOAP* support. Please refer to the *PHP* documentation for more information (<http://www.php.net/>). The files related to this sample are located in `samples/php`.

1 Creating an *APG* graph with *PHP*

The first example is the generation of a graph image using a *PHP* page. Actually, the *PHP* itself will not generate the image but will simply forwards the one generated by the *APG* Web Service (just like our *Java* example).

1.1 Overview

To customize the graph content, our page (`graph.php`) accepts some parameters:

- `filter` is the filter expression to select which metric we want to graph (defaults to '*nothing selected*')

- width, height, and scale which are respectively the width (defaults to 500), the height (defaults to 300) and the scaling factor of the graph (default to 1.0).
- start and end which is the selected time range. These parameters accepts the following values:
 - if end is a positive value, it represents the end date of the selected range
 - if end is null or negative, the actual end timestamp will be *now - end*
 - if start is a positive value, it represents the start date of the selected range
 - if start is omitted or zero, it defaults to *end - 1 day*
 - if start is negative, the actual start timestamp will be *end + start*

For example, if you set end=0 and start=-43200, you will graph the last 12 hours.

Then, the *PHP* page has to create the *XML* Tree template to generate a graph report, retrieve and forward the graph image from the response.

1.2 Report Generation

Using the *PHP SOAP* library is pretty straightforward. You create the request, call the desired method on a *SoapClient* object and explore the response. Here is the report generation request to plot a graph:

```
// create the SoapClient object using the WSDL location and the service credentials
$client = new SoapClient(REP_WS_WSDL, array('login' => WS_USER, 'password' => WS_PASSWORD));
// setup some report properties
$properties = array(
    'property' => array(
        // set the graph width
        array('key' => 'apg.ws.report.graph.width', '_' => $width),
        // set the graph height
        array('key' => 'apg.ws.report.graph.height', '_' => $height),
        // set the graph scaling factor
        array('key' => 'apg.ws.report.graph.scale', '_' => $scale),
        // we want a graph image...
        array('key' => 'apg.ws.report.graph.with.image', '_' => 'true'),
        // ... with the legend inside
        array('key' => 'apg.ws.report.graph.with.legend.image', '_' => 'true')
    )
);
// create an XML Tree template
```

```
$node = array(
    'property' => array(
        new SoapVar(
            // set the filter expression
            array('filterExpression' => $filter),
            XSD_ANYTYPE, 'NodeFilter', XML_TREE_NS),
        new SoapVar(
            // set the report mode
            array('defaultMode' => 'nrx'),
            XSD_ANYTYPE, 'ReportPreferences', XML_TREE_NS),
        new SoapVar(
            // set the selected time range
            array('timeRangeExpression' => 'r:' . $start . ':' . $end . ':0'),
            XSD_ANYTYPE, 'RuntimePreferences', XML_TREE_NS)
    )
);
// call the service and convert objects to array
$response = objtoarray($client->getReport(array(
    'properties' => $properties,
    'node' => $node
)));
// forward graph data to the client
print($response['graph-element']['graph']);
```

2 Simple Report Page

In order to wrap our previous example, we create a simple report page which will take property values as parameters, plot the graph of the selected metrics along with some information on the selected devices.

2.1 Creation of the filter

Our report page uses url parameters to create filters. For example, if we go to <http://localhost/report.php?device=dev1&part=eth0>, we expect the filter to be `device=='dev1' & part=='eth0'`. The only detail you should take care of is to correctly escape the filter patterns. Standard SQL escaping functions should do the job (you can see `make_filter` for more details).

```
// parse parameters and create filter
foreach($_GET as $param => $value) {
```

```
        if(strlen($filter) > 0) {
            $filter = $filter . ' & ';
        }
        $filter = $filter . make_filter($param, $value);
    }

    // if nothing selected, defaults to nothing...
    if(strlen($filter) == 0) {
        $filter = '!(*)';
    }
}
```

2.2 Retrieving device name(s)

To write a title to the page, we retrieve the device name(s) which have been selected. Since the filter may select many devices, we limit the number of selected values to 3 and concatenate device names in a comma separated string.

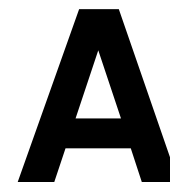
```
// create the SoapClient object using the WSDL location and the service credentials
$client = new SoapClient(DB_WS_WSDL, array('login' => WS_USER, 'password' => WS_PASSWORD));
// call the service and convert objects to array
$ret = objtoarray($client->getDistinctPropertyValues(array(
    'filter' => $filter,
    'property' => 'device',
    'limit' => 3
)));
// retrieve device name values
$values = $ret['distinct-properties']['value'];
// values is either an array, a string or undefined
if(is_array($values)) {
    $cnt = 1;
    foreach($values as $v) {
        if($cnt > 1) {
            $title = $title . ', ';
        }
        if($cnt++ > 2) {
            $title = $title . '...';
            break;
        } else {
            $title = $title . $v;
        }
    }
}
```

```
} else {  
    $title = $values;  
}
```

2.3 Putting all together

With the filter and the title we have just built, we can create our final markup:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  
<html>  
<head>  
    <title><?php echo $title ?></title>  
</head>  
<body>  
    <h1><?php echo $title ?></h1>  
    <p></p>  
</body>  
</html>
```

***DatabaseAccessor* Definition**

1 Performance Data Access

1.1 getObjectData

*Timeseries** **getObjectData**(*filter?*, *sub-filter**, *start-timestamp*, *end-timestamp*, *time-zone?*, *time-filter?*, *period*, *field**, *selected-variable**, *limit?*) throws `Exception`

Retrieves performance data directly from the database. This method doesn't perform any kind of aggregation or timestamp alignment when the database schema allows pure time series queries.

Parameters:

- *filter?*: the base filter expression (optional)
- *sub-filter**: the sub filters, that will be appended to the base filter to select data (zero or more)
- *start-timestamp*: the lowest timestamp of the selected range (required)
- *end-timestamp*: the highest timestamp of the selected range (required)

- *time-zone?*: the time zone of the time range (optional)
- *time-filter?*: a timefilter expression (please refer to XmlTree1.xsd for syntax) (optional)
- *period*: the selected aggregate period (0 means raw values) (required)
- *field**: the selected fields (average, ...). It defaults to AVG. (zero or more)
- *selected-variable**: a collection of object ids, which can be used to restrict selection to a subset of objects (zero or more)
- *limit?*: the maximum number of returned objects (optional)

Returns:

- *timeseries**: one time serie collection per filter/sub filter (zero or more)

Throws:

- *Exception*: forwards any underlying exception

Request Samples:

```
<getObjectData>
  <!--Optional:-->
  <filter?></filter>
  <!--Zero or more repetitions:-->
  <sub-filter?></sub-filter>
  <start-timestamp?></start-timestamp>
  <end-timestamp?></end-timestamp>
  <!--Optional:-->
  <time-filter?></time-filter>
  <period?></period>
  <!--Zero or more repetitions:-->
  <field?></field>
  <!--Zero or more repetitions:-->
  <selected-variable?></selected-variable>
  <!--Optional:-->
  <limit?></limit>
</getObjectData>
```

Response Samples:

```

<getObjectDataResponse>
  <timeseries>
    <timeserie fields="2" id="APG-DB:group-5373" length="2">
      <tv t="1205330400" v="100.0_100.0"/>
      <tv t="1205334000" v="100.0_100.0"/>
    </timeserie>
    <timeserie fields="2" id="APG-DB:group-5476" length="2">
      <tv t="1205330400" v="0.0_0.0"/>
      <tv t="1205334000" v="0.0_0.0"/>
    </timeserie>
  </timeseries>
  <timeseries>
    <timeserie fields="2" id="APG-DB:group-5382" length="2">
      <tv t="1205330400" v="96474.0_96474.0"/>
      <tv t="1205334000" v="96474.0_96474.0"/>
    </timeserie>
    <timeserie fields="2" id="APG-DB:group-5381" length="2">
      <tv t="1205330400" v="96607.1_96598.0"/>
      <tv t="1205334000" v="96611.0_96601.0"/>
    </timeserie>
  </timeseries>
</getObjectDataResponse>

```

1.2 getAggregatedData

Timeserie* **getAggregatedData**(*filter?*, *sub-filter**, *start-timestamp*, *end-timestamp*, *time-zone?*, *time-filter?*, *period*, *aggregations?*) throws Exception

Retrieves aggregated and/or aligned data from database. Contrary to getGraphData, this method return only one dataset per filter/sub filter. If more that one object is selected, the values will be aggregated. Returned dataset values are always aligned on the nearest period multiple.

Parameters:

- *filter?*: the base filter expression (optional)
- *sub-filter**: the sub filters, that will be appended to the base filter to select data (zero or more)

- *start-timestamp*: the lowest timestamp of the selected range (required)
- *end-timestamp*: the highest timestamp of the selected range (required)
- *time-zone?*: the time zone of the time range (optional)
- *time-filter?*: a timefilter expression (please refer to XmlTree1.xsd for syntax) (optional)
- *period*: the aggregation period (required)
- *aggregations?*: the multiple aggregation functions to use (count aggregation, spacial aggregation and temporal aggregation). It defaults to spacio-temporal average. (optional)

Returns:

- *timeserie**: one time serie per filter/sub filter (zero or more)

Throws:

- *Exception*: forwards any underlying exception

Request Samples:

```
<getAggregatedData>
  <!--Optional:-->
  <filter>?</filter>
  <!--Zero or more repetitions:-->
  <sub-filter>?</sub-filter>
  <start-timestamp>?</start-timestamp>
  <end-timestamp>?</end-timestamp>
  <!--Optional:-->
  <time-filter>?</time-filter>
  <period>?</period>
  <!--Optional:-->
  <aggregations count="?" spacial="?" temporal="?" />
</getAggregatedData>
```

Response Samples:

```
<getAggregatedDataResponse>
  <timeserie fields="2" length="3">
    <tv t="1205326800" v="12090.652399619124_57.0"/>
    <tv t="1205330400" v="9992.277105933043_57.0"/>
    <tv t="1205334000" v="12902.927867071307_57.0"/>
  </timeserie>
  <timeserie fields="2" length="3">
    <tv t="1205326800" v="0.0_5.0"/>
    <tv t="1205330400" v="0.0_5.0"/>
    <tv t="1205334000" v="0.0_5.0"/>
  </timeserie>
</getAggregatedDataResponse>
```

2 Topology Data Access

2.1 getObjectProperties

Object-properties* **getObjectProperties**(*filter?*, *sub-filter**, *start-timestamp?*, *end-timestamp?*, *time-zone?*, *time-filter?*, *property**) throws Exception

Retrieves topology data directly from the database for each selected object. This method doesn't perform any kind of processing.

Parameters:

- *filter?*: the base filter expression (optional)
- *sub-filter**: the sub filters, that will be appended to the base filter to select data (zero or more)
- *start-timestamp?*: the lowest timestamp of the selected range (optional)
- *end-timestamp?*: the highest timestamp of the selected range (optional)
- *time-zone?*: the time zone of the time range (optional)
- *time-filter?*: a timefilter expression (please refer to XmlTree1.xsd for syntax) (optional)
- *property**: the list of property names (zero or more)

Returns:

- *object-properties**: one property collection per filter/sub filter (zero or more)

Throws:

- *Exception*: forwards any underlying exception

Request Samples:

```
<getObjectProperties>
  <!--Optional:-->
  <filter>?</filter>
  <!--Zero or more repetitions:-->
  <sub-filter>?</sub-filter>
  <!--Zero or more repetitions:-->
  <property>?</property>
</getObjectProperties>
```

Response Samples:

```
<getObjectPropertiesResponse>
  <object-properties>
    <properties id="APG-DB:group-5373">
      <value property="device">w4n-rtr01</value>
      <value property="part">FastEthernet1</value>
    </properties>
    <properties id="APG-DB:group-5471">
      <value property="device">w4n-rtr01</value>
      <value property="part">FastEthernet0</value>
    </properties>
  </object-properties>
  <object-properties>
    <properties id="APG-DB:group-8032">
      <value property="device">al-dual</value>
      <value property="part">Physical Memory</value>
    </properties>
    <properties id="APG-DB:group-6137">
```

```
<value property="device">al-dual</value>
<value property="part">Virtual Memory</value>
</properties>
</object-properties>
</getObjectPropertiesResponse>
```

2.2 getDistinctPropertyValues

Distinct-properties* **getDistinctPropertyValues**(*filter?*, *sub-filter**, *start-timestamp?*, *end-timestamp?*, *time-zone?*, *time-filter?*, *property**, *limit?*) throws Exception

Returns distinct property values from the database. When more than one property is specified, the returned collected are the union of all the specified property values.

Parameters:

- *filter?*: the base filter expression (optional)
- *sub-filter**: the sub filters, that will be appended to the base filter to select data (zero or more)
- *start-timestamp?*: the lowest timestamp of the selected range (optional)
- *end-timestamp?*: the highest timestamp of the selected range (optional)
- *time-zone?*: the time zone of the time range (optional)
- *time-filter?*: a timefilter expression (please refer to XmlTree1.xsd for syntax) (optional)
- *property**: the list of property names (zero or more)
- *limit?*: the maximum number of retrieves values (optional)

Returns:

- *distinct-properties**: one distinct property collection per filter/sub filter (zero or more)

Throws:

- *Exception*: forwards any underlying exception

Request Samples:

```
<getDistinctPropertyValues>
  <!--Optional:-->
  <filter>?</filter>
  <!--Zero or more repetitions:-->
  <sub-filter>?</sub-filter>
  <!--Zero or more repetitions:-->
  <property>?</property>
  <!--Optional:-->
  <limit>?</limit>
</getDistinctPropertyValues>
```

Response Samples:

```
<getDistinctPropertyValuesResponse>
  <distinct-properties>
    <value>CurrentUtilization</value>
  </distinct-properties>
  <distinct-properties>
    <value>StorageSize</value>
    <value>AvailableSpace</value>
  </distinct-properties>
</getDistinctPropertyValuesResponse>
```

2.3 getDistinctPropertyRecords

Record* **getDistinctPropertyRecords**(*filter?*, *start-timestamp?*, *end-timestamp?*, *time-zone?*, *time-filter?*, *property**) throws Exception

Returns all the distinct property value combination from the database. For example, if you request property records for properties p1 and p2, you will get all the existing combination (v1_1,v2_1), (v1_1,v2_2), (v1_1,v2_3), ...

Parameters:

- *filter?*: the filter expression (optional)
- *start-timestamp?*: the lowest timestamp of the selected range (optional)
- *end-timestamp?*: the highest timestamp of the selected range (optional)
- *time-zone?*: the time zone of the time range (optional)
- *time-filter?*: a timefilter expression (please refer to XmlTree1.xsd for syntax) (optional)
- *property**: the list of property names (zero or more)

Returns:

- *record**: some property records (a property record is a combination of property values) (zero or more)

Throws:

- *Exception*: forwards any underlying exception

Request Samples:

```
<getDistinctPropertyRecords>
  <!--Optional:-->
  <filter?></filter>
  <!--Zero or more repetitions:-->
  <property?></property>
</getDistinctPropertyRecords>
```

Response Samples:

```
<getDistinctPropertyRecordsResponse>
  <record>
    <value>FileSystem</value>
    <value>AvailableSpace</value>
  </record>
  <record>
    <value>FileSystem</value>
```

```
<value>StorageSize</value>
</record>
</getDistinctPropertyRecordsResponse>
```

3 Topology Base Object Access

3.1 getObjectFilters

Filter* **getObjectFilters**(*filter?*, *start-timestamp*, *end-timestamp*, *time-zone?*, *time-filter?*, *limit?*) throws Exception

Returns the filters required to retrieve performance or topology data from individual objects selected by a filter. Depending on the database schema, the timestamps may not be used.

Parameters:

- *filter?*: the filter expression (optional)
- *start-timestamp*: the lowest timestamp of the selected range (required)
- *end-timestamp*: the highest timestamp of the selected range (required)
- *time-zone?*: the time zone of the time range (optional)
- *time-filter?*: a timefilter expression (please refer to XmlTree1.xsd for syntax) (optional)
- *limit?*: the maximum number of objects to return (optional)

Returns:

- *filter**: the filters which select individual objects (zero or more)

Throws:

- *Exception*: forwards any underlying exception

Request Samples:

```
<getObjectFilters>
  <!--Optional:-->
  <filter>?</filter>
  <start-timestamp>?</start-timestamp>
  <end-timestamp>?</end-timestamp>
  <!--Optional:-->
  <time-filter>?</time-filter>
  <!--Optional:-->
  <limit>?</limit>
</getObjectFilters>
```

Response Samples:

```
<getObjectFiltersResponse>
  <filter>#APG-DB:group-8673</filter>
  <filter>#APG-DB:group-8674</filter>
</getObjectFiltersResponse>
```

3.2 getObjectCount

Count **getObjectCount**(*filter?*, *start-timestamp*, *end-timestamp*, *time-zone?*, *time-filter?*, *limit?*) throws Exception

Returns the number of base object selected by a filter. Depending on the database schema, the timestamps may not be used.

Parameters:

- *filter?*: the filter expression (optional)
- *start-timestamp*: the lowest timestamp of the selected range (required)
- *end-timestamp*: the highest timestamp of the selected range (required)
- *time-zone?*: the time zone of the time range (optional)

- *time-filter?*: a timefilter expression (please refer to XmlTree1.xsd for syntax) (optional)
- *limit?*: the maximum number of objects to return (optional)

Returns:

- *count*: the number of base object selected by the filter (required)

Throws:

- *Exception*: forwards any underlying exception

Request Samples:

```
<getObjectCount>
  <!--Optional:-->
  <filter>?</filter>
  <start-timestamp>?</start-timestamp>
  <end-timestamp>?</end-timestamp>
  <!--Optional:-->
  <time-filter>?</time-filter>
  <!--Optional:-->
  <limit>?</limit>
</getObjectCount>
```

Response Samples:

```
<getObjectCountResponse>
  <count>2</count>
</getObjectCountResponse>
```

3.3 getStaticObjectFilters

Filter* **getStaticObjectFilters**(*filter?*) throws Exception

This is the same method as getObjectFilters but without temporal information. Depending on the database schema, this may have the same effect.

Parameters:

- *filter?*: the filter expression (optional)

Returns:

- *filter**: the filters which select individual objects (zero or more)

Throws:

- *Exception*: forwards any underlying exception

Request Samples:

```
<getStaticObjectFilters>
  <!--Optional:-->
  <filter>?</filter>
</getStaticObjectFilters>
```

Response Samples:

```
<getStaticObjectFiltersResponse>
  <filter>#APG-DB:group-5368</filter>
  <filter>#APG-DB:group-5462</filter>
</getStaticObjectFiltersResponse>
```

3.4 getStaticObjectCount

Count **getStaticObjectCount**(*filter?*) throws Exception

This is the same method as getObjectFiltersCount but without temporal information. Depending on the database schema, this may have the same effect.

Parameters:

- *filter?*: the filter expression (optional)

Returns:

- *count*: the number of base object selected by the filter (required)

Throws:

- *Exception*: forwards any underlying exception

Request Samples:

```
<getStaticObjectCount>
  <!--Optional:-->
  <filter?></filter>
</getStaticObjectCount>
```

Response Samples:

```
<getStaticObjectCountResponse>
  <count>3375</count>
</getStaticObjectCountResponse>
```

4 Schema Metadata Access

4.1 getAvailableAccessors

Accessor* **getAvailableAccessors**(*filter?*) throws Exception

Returns the names of the database accessors currently running. An accessor is uniquely defined by a string formatted as TYPE-NAME (e.g. APG-DB1). A filter can be specified to restrict returned values. However, these filters are not evaluated in database and should only contain database selection predicates (like #APG:ALL — #RPE:ALL).

Parameters:

- *filter?*: the filter expression (optional)

Returns:

- *accessor**: the available accessor names (zero or more)

Throws:

- *Exception*: forwards any underlying exception

Request Samples:

```
<getAvailableAccessors>
  <!--Optional:-->
  <filter?></filter>
</getAvailableAccessors>
```

Response Samples:

```
<getAvailableAccessorsResponse>
  <accessor>APG-DB</accessor>
</getAvailableAccessorsResponse>
```

4.2 getAvailableAggregatePeriods

Period* **getAvailableAggregatePeriods**(*filter?*) throws Exception

Returns the aggregate periods that are available from database. For time series schemas, this is the list of period that can be accessed by `getGraphData`. A filter can be specified to restrict returned values. However, these filters are not evaluated in database and should only contain database selection predicates (like `#APG:ALL` — `#RPE:ALL`).

Parameters:

- *filter?*: the filter expression (optional)

Returns:

- *period**: the list of available aggregation periods (zero or more)

Throws:

- *Exception*: forwards any underlying exception

Request Samples:

```
<getAvailableAggregatePeriods>
  <!--Optional:-->
  <filter?></filter>
</getAvailableAggregatePeriods>
```

Response Samples:

```
<getAvailableAggregatePeriodsResponse>
  <period>0</period>
  <period>600</period>
  <period>1800</period>
  <period>3600</period>
  <period>7200</period>
  <period>86400</period>
  <period>604800</period>
</getAvailableAggregatePeriodsResponse>
```

4.3 getAvailableProperties

Property* **getAvailableProperties**(*filter?*) throws Exception

Returns the description of available properties.

Parameters:

- *filter?*: the filter expression (optional)

Returns:

- *property**: the available property descriptions (zero or more)

Throws:

- *Exception*: forwards any underlying exception

Request Samples:

```
<getAvailableProperties>
  <!--Optional:-->
  <filter?></filter>
</getAvailableProperties>
```

Response Samples:

```
<getAvailablePropertiesResponse>
  <property accessor="APG-DB" name="device">The name or address of the device.</property>
  <property accessor="APG-DB" name="source">The source of the metric (typically an EMC Smarts
    domain name, a collector identifier, ...).</property>
</getAvailablePropertiesResponse>
```

4.4 getSuggestions

Suggestion* **getSuggestions**(*filter?*, *pattern?*, *property**, *limit?*) throws Exception

Retrieves property values suggestions based on some criteria. It retrieves suggestions for the specified pattern if any. The suggestions can be restricted using a filter or a set of properties to inspect. The limit is used to restrict the maximum number of values returned by the call.

Parameters:

- *filter?*: a refinement filter expression (optional)
- *pattern?*: a SQL like pattern (optional)

- *property**: a list of property names to search values from (zero or more)
- *limit?*: the maximum number of returned suggestions (optional)

Returns:

- *suggestion**: a set of suggestions (zero or more)

Throws:

- *Exception*: forwards any underlying exception

Request Samples:

```
<getSuggestions>
  <!--Optional:-->
  <filter?></filter>
  <!--Optional:-->
  <pattern?></pattern>
  <!--Zero or more repetitions:-->
  <property?></property>
  <!--Optional:-->
  <limit?></limit>
</getSuggestions>
```

Response Samples:

```
<getSuggestionsResponse>
  <suggestion accessor="APG-DB" property="location">Montreal</suggestion>
  <suggestion accessor="APG-DB" property="city">Montreal</suggestion>
</getSuggestionsResponse>
```

5 WSDL Service Definition

This section includes the relevant *WSDL* and *XSD* files related to the *DatabaseAccessor* service definition. It can be useful to see the exact definition of an operation or a parameter using the included files.

5.1 DatabaseAccessor.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.4.1-
hudson-346-. --><!--Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-
WS RI 2.1.4.1-hudson-346-. -->
<definitions targetNamespace="http://www.watch4net.com/APG/Remote/DatabaseAccessorService" name
="DatabaseAccessorService"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.
xsd" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://www.watch4net.com/APG/Remote/DatabaseAccessorService" xmlns:xsd="http://
www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types>
        <xsd:schema>
            <xsd:import namespace="http://www.watch4net.com/APG/Remote/DatabaseAccessorService"
schemaLocation="DatabaseAccessor.xsd" />
        </xsd:schema>
    </types>
    <message name="getObjectData">
        <part name="parameters" element="tns:getObjectData" />
    </message>
    <message name="getObjectDataResponse">
        <part name="parameters" element="tns:getObjectDataResponse" />
    </message>
    <message name="Exception">
        <part name="fault" element="tns:Exception" />
    </message>
    <message name="getAggregatedData">
        <part name="parameters" element="tns:getAggregatedData" />
    </message>
    <message name="getAggregatedDataResponse">
        <part name="parameters" element="tns:getAggregatedDataResponse" />
    </message>
    <message name="getObjectProperties">
        <part name="parameters" element="tns:getObjectProperties" />
    </message>
    <message name="getObjectPropertiesResponse">
        <part name="parameters" element="tns:getObjectPropertiesResponse" />
    </message>
    <message name="getDistinctPropertyValues">
        <part name="parameters" element="tns:getDistinctPropertyValues" />
    </message>
```

```
</message>
<message name="getDistinctPropertyValuesResponse">
  <part name="parameters" element="tns:getDistinctPropertyValuesResponse" />
</message>
<message name="getDistinctPropertyRecords">
  <part name="parameters" element="tns:getDistinctPropertyRecords" />
</message>
<message name="getDistinctPropertyRecordsResponse">
  <part name="parameters" element="tns:getDistinctPropertyRecordsResponse" />
</message>
<message name="getObjectFilters">
  <part name="parameters" element="tns:getObjectFilters" />
</message>
<message name="getObjectFiltersResponse">
  <part name="parameters" element="tns:getObjectFiltersResponse" />
</message>
<message name="getObjectCount">
  <part name="parameters" element="tns:getObjectCount" />
</message>
<message name="getObjectCountResponse">
  <part name="parameters" element="tns:getObjectCountResponse" />
</message>
<message name="getStaticObjectFilters">
  <part name="parameters" element="tns:getStaticObjectFilters" />
</message>
<message name="getStaticObjectFiltersResponse">
  <part name="parameters" element="tns:getStaticObjectFiltersResponse" />
</message>
<message name="getStaticObjectCount">
  <part name="parameters" element="tns:getStaticObjectCount" />
</message>
<message name="getStaticObjectCountResponse">
  <part name="parameters" element="tns:getStaticObjectCountResponse" />
</message>
<message name="getAvailableAccessors">
  <part name="parameters" element="tns:getAvailableAccessors" />
</message>
<message name="getAvailableAccessorsResponse">
  <part name="parameters" element="tns:getAvailableAccessorsResponse" />
</message>
<message name="getAvailableAggregatePeriods">
  <part name="parameters" element="tns:getAvailableAggregatePeriods" />
```

```
</message>
<message name="getAvailableAggregatePeriodsResponse">
  <part name="parameters" element="tns:getAvailableAggregatePeriodsResponse" />
</message>
<message name="getAvailableProperties">
  <part name="parameters" element="tns:getAvailableProperties" />
</message>
<message name="getAvailablePropertiesResponse">
  <part name="parameters" element="tns:getAvailablePropertiesResponse" />
</message>
<message name="getSuggestions">
  <part name="parameters" element="tns:getSuggestions" />
</message>
<message name="getSuggestionsResponse">
  <part name="parameters" element="tns:getSuggestionsResponse" />
</message>
<portType name="DatabaseAccessor">
  <operation name="getObjectData">
    <input message="tns:getObjectData" />
    <output message="tns:getObjectDataResponse" />
    <fault message="tns:Exception" name="Exception" />
  </operation>
  <operation name="getAggregatedData">
    <input message="tns:getAggregatedData" />
    <output message="tns:getAggregatedDataResponse" />
    <fault message="tns:Exception" name="Exception" />
  </operation>
  <operation name="getObjectProperties">
    <input message="tns:getObjectProperties" />
    <output message="tns:getObjectPropertiesResponse" />
    <fault message="tns:Exception" name="Exception" />
  </operation>
  <operation name="getDistinctPropertyValues">
    <input message="tns:getDistinctPropertyValues" />
    <output message="tns:getDistinctPropertyValuesResponse" />
    <fault message="tns:Exception" name="Exception" />
  </operation>
  <operation name="getDistinctPropertyRecords">
    <input message="tns:getDistinctPropertyRecords" />
    <output message="tns:getDistinctPropertyRecordsResponse" />
    <fault message="tns:Exception" name="Exception" />
  </operation>
</portType>
```

```
<operation name="getObjectFilters">
  <input message="tns:getObjectFilters" />
  <output message="tns:getObjectFiltersResponse" />
  <fault message="tns:Exception" name="Exception" />
</operation>
<operation name="getObjectCount">
  <input message="tns:getObjectCount" />
  <output message="tns:getObjectCountResponse" />
  <fault message="tns:Exception" name="Exception" />
</operation>
<operation name="getStaticObjectFilters">
  <input message="tns:getStaticObjectFilters" />
  <output message="tns:getStaticObjectFiltersResponse" />
  <fault message="tns:Exception" name="Exception" />
</operation>
<operation name="getStaticObjectCount">
  <input message="tns:getStaticObjectCount" />
  <output message="tns:getStaticObjectCountResponse" />
  <fault message="tns:Exception" name="Exception" />
</operation>
<operation name="getAvailableAccessors">
  <input message="tns:getAvailableAccessors" />
  <output message="tns:getAvailableAccessorsResponse" />
  <fault message="tns:Exception" name="Exception" />
</operation>
<operation name="getAvailableAggregatePeriods">
  <input message="tns:getAvailableAggregatePeriods" />
  <output message="tns:getAvailableAggregatePeriodsResponse" />
  <fault message="tns:Exception" name="Exception" />
</operation>
<operation name="getAvailableProperties">
  <input message="tns:getAvailableProperties" />
  <output message="tns:getAvailablePropertiesResponse" />
  <fault message="tns:Exception" name="Exception" />
</operation>
<operation name="getSuggestions">
  <input message="tns:getSuggestions" />
  <output message="tns:getSuggestionsResponse" />
  <fault message="tns:Exception" name="Exception" />
</operation>
</portType>
<binding name="DatabaseAccessorPortBinding" type="tns:DatabaseAccessor">
```

```
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
<operation name="getObjectData">
  <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
  <fault name="Exception">
    <soap:fault name="Exception" use="literal" />
  </fault>
</operation>
<operation name="getAggregatedData">
  <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
  <fault name="Exception">
    <soap:fault name="Exception" use="literal" />
  </fault>
</operation>
<operation name="getObjectProperties">
  <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
  <fault name="Exception">
    <soap:fault name="Exception" use="literal" />
  </fault>
</operation>
<operation name="getDistinctPropertyValues">
  <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
```

```
<output>
  <soap:body use="literal" />
</output>
<fault name="Exception">
  <soap:fault name="Exception" use="literal" />
</fault>
</operation>
<operation name="getDistinctPropertyRecords">
  <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
  <fault name="Exception">
    <soap:fault name="Exception" use="literal" />
  </fault>
</operation>
<operation name="getObjectFilters">
  <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
  <fault name="Exception">
    <soap:fault name="Exception" use="literal" />
  </fault>
</operation>
<operation name="getObjectCount">
  <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
  <fault name="Exception">
    <soap:fault name="Exception" use="literal" />
  </fault>
</operation>
```

```
</operation>
<operation name="getStaticObjectFilters">
  <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
  <fault name="Exception">
    <soap:fault name="Exception" use="literal" />
  </fault>
</operation>
<operation name="getStaticObjectCount">
  <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
  <fault name="Exception">
    <soap:fault name="Exception" use="literal" />
  </fault>
</operation>
<operation name="getAvailableAccessors">
  <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
  <fault name="Exception">
    <soap:fault name="Exception" use="literal" />
  </fault>
</operation>
<operation name="getAvailableAggregatePeriods">
  <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
```

```
<output>
  <soap:body use="literal" />
</output>
<fault name="Exception">
  <soap:fault name="Exception" use="literal" />
</fault>
</operation>
<operation name="getAvailableProperties">
  <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
  <fault name="Exception">
    <soap:fault name="Exception" use="literal" />
  </fault>
</operation>
<operation name="getSuggestions">
  <soap:operation soapAction="" />
  <input>
    <soap:body use="literal" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
  <fault name="Exception">
    <soap:fault name="Exception" use="literal" />
  </fault>
</operation>
</binding>
<service name="DatabaseAccessorService">
  <port name="DatabaseAccessorPort" binding="tns:DatabaseAccessorPortBinding">
    <soap:address location="http://localhost:58080/APG-WS/wsapi/db" />
  </port>
</service>
</definitions>
```

5.2 DatabaseAccessor.xsd

```

<?xml version="1.0" encoding="UTF-8"?>
<!--Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.4.1-
hudson-346-. -->
<xs:schema version="1.0" targetNamespace="http://www.watch4net.com/APG/Remote/
DatabaseAccessorService"
  xmlns:tns="http://www.watch4net.com/APG/Remote/DatabaseAccessorService" xmlns:xs="http://
www.w3.org/2001/XMLSchema">
  <xs:element name="Exception" type="tns:Exception" />
  <xs:element name="getAggregatedData" type="tns:getAggregatedData" />
  <xs:element name="getAggregatedDataResponse" type="tns:getAggregatedDataResponse" />
  <xs:element name="getAvailableAccessors" type="tns:getAvailableAccessors" />
  <xs:element name="getAvailableAccessorsResponse" type="tns:getAvailableAccessorsResponse" /
>
  <xs:element name="getAvailableAggregatePeriods" type="tns:getAvailableAggregatePeriods" />
  <xs:element name="getAvailableAggregatePeriodsResponse" type="
tns:getAvailableAggregatePeriodsResponse" />
  <xs:element name="getAvailableProperties" type="tns:getAvailableProperties" />
  <xs:element name="getAvailablePropertiesResponse" type="tns:getAvailablePropertiesResponse"
/>
  <xs:element name="getDistinctPropertyRecords" type="tns:getDistinctPropertyRecords" />
  <xs:element name="getDistinctPropertyRecordsResponse" type="
tns:getDistinctPropertyRecordsResponse" />
  <xs:element name="getDistinctPropertyValues" type="tns:getDistinctPropertyValues" />
  <xs:element name="getDistinctPropertyValuesResponse" type="
tns:getDistinctPropertyValuesResponse" />
  <xs:element name="getObjectCount" type="tns:getObjectCount" />
  <xs:element name="getObjectCountResponse" type="tns:getObjectCountResponse" />
  <xs:element name="getObjectData" type="tns:getObjectData" />
  <xs:element name="getObjectDataResponse" type="tns:getObjectDataResponse" />
  <xs:element name="getObjectFilters" type="tns:getObjectFilters" />
  <xs:element name="getObjectFiltersResponse" type="tns:getObjectFiltersResponse" />
  <xs:element name="getObjectProperties" type="tns:getObjectProperties" />
  <xs:element name="getObjectPropertiesResponse" type="tns:getObjectPropertiesResponse" />
  <xs:element name="getStaticObjectCount" type="tns:getStaticObjectCount" />
  <xs:element name="getStaticObjectCountResponse" type="tns:getStaticObjectCountResponse" />
  <xs:element name="getStaticObjectFilters" type="tns:getStaticObjectFilters" />
  <xs:element name="getStaticObjectFiltersResponse" type="tns:getStaticObjectFiltersResponse"
/>
  <xs:element name="getSuggestions" type="tns:getSuggestions" />
  <xs:element name="getSuggestionsResponse" type="tns:getSuggestionsResponse" />
  <xs:complexType name="getAvailableProperties">

```

```

        <xs:sequence>
          <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" />
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="getAvailablePropertiesResponse">
        <xs:sequence>
          <xs:element name="property" type="tns:Property" form="qualified" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="Property">
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="accessor" type="xs:string" use="required" />
            <xs:attribute name="name" type="xs:string" use="required" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
      <xs:complexType name="Exception">
        <xs:sequence>
          <xs:element name="message" type="xs:string" form="qualified" minOccurs="0" />
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="getStaticObjectCount">
        <xs:sequence>
          <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" />
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="getStaticObjectCountResponse">
        <xs:sequence>
          <xs:element name="count" type="xs:int" form="qualified" />
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="getObjectFilters">
        <xs:sequence>
          <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" />
          <xs:element name="start-timestamp" type="xs:int" form="qualified" />
          <xs:element name="end-timestamp" type="xs:int" form="qualified" />
          <xs:element name="time-zone" type="xs:string" form="qualified" minOccurs="0" />
          <xs:element name="time-filter" type="xs:string" form="qualified" minOccurs="0" />
          <xs:element name="limit" type="xs:int" form="qualified" minOccurs="0" />
        </xs:sequence>
      </xs:complexType>

```

```

</xs:complexType>
<xs:complexType name="getObjectFiltersResponse">
  <xs:sequence>
    <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" maxOccurs=
"unbounded" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="getObjectCount">
  <xs:sequence>
    <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" />
    <xs:element name="start-timestamp" type="xs:int" form="qualified" />
    <xs:element name="end-timestamp" type="xs:int" form="qualified" />
    <xs:element name="time-zone" type="xs:string" form="qualified" minOccurs="0" />
    <xs:element name="time-filter" type="xs:string" form="qualified" minOccurs="0" />
    <xs:element name="limit" type="xs:int" form="qualified" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="getObjectCountResponse">
  <xs:sequence>
    <xs:element name="count" type="xs:int" form="qualified" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="getStaticObjectFilters">
  <xs:sequence>
    <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="getStaticObjectFiltersResponse">
  <xs:sequence>
    <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" maxOccurs=
"unbounded" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="getDistinctPropertyRecords">
  <xs:sequence>
    <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" />
    <xs:element name="start-timestamp" type="xs:int" form="qualified" minOccurs="0" />
    <xs:element name="end-timestamp" type="xs:int" form="qualified" minOccurs="0" />
    <xs:element name="time-zone" type="xs:string" form="qualified" minOccurs="0" />
    <xs:element name="time-filter" type="xs:string" form="qualified" minOccurs="0" />
    <xs:element name="property" type="xs:string" form="qualified" minOccurs="0"
maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

```

```

        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="getDistinctPropertyRecordsResponse">
        <xs:sequence>
            <xs:element name="record" type="tns:PropertyRecord" form="qualified" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="PropertyRecord">
        <xs:sequence>
            <xs:element name="value" type="xs:string" form="qualified" minOccurs="0" maxOccurs="
unbounded" />
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="getSuggestions">
        <xs:sequence>
            <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" />
            <xs:element name="pattern" type="xs:string" form="qualified" minOccurs="0" />
            <xs:element name="property" type="xs:string" form="qualified" minOccurs="0"
maxOccurs="unbounded" />
            <xs:element name="limit" type="xs:int" form="qualified" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="getSuggestionsResponse">
        <xs:sequence>
            <xs:element name="suggestion" type="tns:Suggestion" form="qualified" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Suggestion" final="extension_restriction">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="accessor" type="xs:string" use="required" />
                <xs:attribute name="property" type="xs:string" use="required" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="getObjectProperties">
        <xs:sequence>
            <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" />
            <xs:element name="sub-filter" type="xs:string" form="qualified" nillable="true"
minOccurs="0" maxOccurs="unbounded" />

```

```

        <xs:element name="start-timestamp" type="xs:int" form="qualified" minOccurs="0" />
        <xs:element name="end-timestamp" type="xs:int" form="qualified" minOccurs="0" />
        <xs:element name="time-zone" type="xs:string" form="qualified" minOccurs="0" />
        <xs:element name="time-filter" type="xs:string" form="qualified" minOccurs="0" />
        <xs:element name="property" type="xs:string" form="qualified" minOccurs="0"
maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="getObjectPropertiesResponse">
    <xs:sequence>
        <xs:element name="object-properties" type="tns:ObjectPropertyValues" form="qualified"
" nillable="true"
            minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="ObjectPropertyValues">
    <xs:sequence>
        <xs:element name="properties" type="tns:PropertyValues" form="qualified" minOccurs="
0" maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="PropertyValues">
    <xs:sequence>
        <xs:element name="value" type="tns:PropertyValue" form="qualified" minOccurs="0"
maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" />
</xs:complexType>
<xs:complexType name="PropertyValue">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="property" type="xs:string" use="required" />
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="getAggregatedData">
    <xs:sequence>
        <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" />
        <xs:element name="sub-filter" type="xs:string" form="qualified" nillable="true"
minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="start-timestamp" type="xs:int" form="qualified" />
        <xs:element name="end-timestamp" type="xs:int" form="qualified" />

```

```

        <xs:element name="time-zone" type="xs:string" form="qualified" minOccurs="0" />
        <xs:element name="time-filter" type="xs:string" form="qualified" minOccurs="0" />
        <xs:element name="period" type="xs:int" form="qualified" />
        <xs:element name="aggregations" type="tns:Aggregations" form="qualified" minOccurs="
0" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="Aggregations">
    <xs:sequence />
    <xs:attribute name="count" type="tns:Aggregation" />
    <xs:attribute name="spacial" type="tns:Aggregation" />
    <xs:attribute name="temporal" type="tns:Aggregation" use="required" />
</xs:complexType>
<xs:complexType name="getAggregatedDataResponse">
    <xs:sequence>
        <xs:element name="timeserie" type="tns:TimeSerie" form="qualified" nillable="true"
minOccurs="0"
            maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="TimeSerie">
    <xs:sequence>
        <xs:element name="tv" type="tns:TimeSerieValue" form="qualified" minOccurs="0"
maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="fields" type="xs:int" use="required" />
    <xs:attribute name="id" type="xs:string" />
    <xs:attribute name="length" type="xs:int" use="required" />
</xs:complexType>
<xs:complexType name="TimeSerieValue">
    <xs:sequence />
    <xs:attribute name="v" use="required">
        <xs:simpleType>
            <xs:list itemType="xs:double" />
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="t" type="xs:int" use="required" />
</xs:complexType>
<xs:complexType name="getAvailableAggregatePeriods">
    <xs:sequence>
        <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" />
    </xs:sequence>

```

```

</xs:complexType>
<xs:complexType name="getAvailableAggregatePeriodsResponse">
  <xs:sequence>
    <xs:element name="period" type="xs:int" form="qualified" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="getDistinctPropertyValues">
  <xs:sequence>
    <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" />
    <xs:element name="sub-filter" type="xs:string" form="qualified" nillable="true" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="start-timestamp" type="xs:int" form="qualified" minOccurs="0" />
    <xs:element name="end-timestamp" type="xs:int" form="qualified" minOccurs="0" />
    <xs:element name="time-zone" type="xs:string" form="qualified" minOccurs="0" />
    <xs:element name="time-filter" type="xs:string" form="qualified" minOccurs="0" />
    <xs:element name="property" type="xs:string" form="qualified" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="limit" type="xs:int" form="qualified" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="getDistinctPropertyValuesResponse">
  <xs:sequence>
    <xs:element name="distinct-properties" type="tns:DistinctPropertyValues" form="qualified" nillable="true" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="DistinctPropertyValues">
  <xs:sequence>
    <xs:element name="value" type="xs:string" form="qualified" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="getAvailableAccessors">
  <xs:sequence>
    <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="getAvailableAccessorsResponse">
  <xs:sequence>

```

```

        <xs:element name="accessor" type="xs:string" form="qualified" minOccurs="0"
maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="getObjectData">
    <xs:sequence>
        <xs:element name="filter" type="xs:string" form="qualified" minOccurs="0" />
        <xs:element name="sub-filter" type="xs:string" form="qualified" nillable="true"
minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="start-timestamp" type="xs:int" form="qualified" />
        <xs:element name="end-timestamp" type="xs:int" form="qualified" />
        <xs:element name="time-zone" type="xs:string" form="qualified" minOccurs="0" />
        <xs:element name="time-filter" type="xs:string" form="qualified" minOccurs="0" />
        <xs:element name="period" type="xs:int" form="qualified" />
        <xs:element name="field" type="tns:Aggregation" form="qualified" nillable="true"
minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="selected-variable" type="xs:string" form="qualified" minOccurs="0"
maxOccurs="unbounded" />
        <xs:element name="limit" type="xs:int" form="qualified" minOccurs="0" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="getObjectDataResponse">
    <xs:sequence>
        <xs:element name="timeseries" type="tns:TimeSeries" form="qualified" nillable="true"
minOccurs="0"
        maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="TimeSeries">
    <xs:sequence>
        <xs:element name="timeserie" type="tns:TimeSerie" form="qualified" minOccurs="0"
maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:simpleType name="Aggregation">
    <xs:restriction base="xs:string">
        <xs:enumeration value="AVG" />
        <xs:enumeration value="MIN" />
        <xs:enumeration value="MAX" />
        <xs:enumeration value="SUM" />
        <xs:enumeration value="LAST" />
        <xs:enumeration value="NBVAL" />
    </xs:restriction>
</xs:simpleType>

```

```
        <xs:enumeration value="LASTTS" />
    </xs:restriction>
</xs:simpleType>
</xs:schema>
```




***ReportManager* Definition**

1 Report Generation Interface

1.1 getReport

Report-element **getReport**(*properties?*, *node*) throws Exception

This function generates a report based on properties and an XML tree fragment. The report properties will modify the report generation/rendering behavior. All the properties are optional and some have a default value. Defaults are primarily read from a private resource file and finally, if `apg.ws.report.defaults` is set to a valid property file name, user defaults are then loaded from there. The XML tree format is exactly the same that the one used in the APG Web Portal. The returned report is always in an XML format.

Parameters:

- *properties?*: some properties (key, values) to modify the behavior of the report generation or rendering (optional)
- *node*: the XML tree fragment root (required)

Returns:

- *report-element*: the root element of the generated XML report (a compound element, a graph element, a table element, an error element or an image element) (required)

Throws:

- *Exception*: forwards any underlying exception

Request Samples:

```
<getReport>
  <properties>
    <property key="apg.ws.mtom">true</property>[...]</properties>
    <node name="Root" singleNodeId="1234">[...]</node>
  </getReport>
```

Response Samples:

```
<getReportResponse>
  <graph-element id="31d664db" name="ftp_ww4n-ds-any01.int.watch4net.net" type="check">
    <report-preferences defaultMode="vrX" duration="3" preferredPeriod="3600"/>
    <graph contentType="image/png">
      <Include href="cid:fb63639d-ac8a-4325-a349-ccfe081ba955@watch4net.com"/>
    </graph>
    <datasets>
      <dataset>
        <dd>
          <d>Timestamp</d>
          <d>LastResponseTime (ms) Min</d>
          <d>LastResponseTime (ms) Max</d>
          <d>LastResponseTime (ms) Average</d>
        </dd>
      </dataset>
    </datasets>
  </graph-element>
</getReportResponse>
```

```

        <d>LastResponseTime (ms) Base Line</d>
    </dd>
    <dv>
        <v>1206478800</v>
        <v>123.467</v>
        <v>330.0</v>
        <v>30.0</v>
        <v>121.85024999999999</v>
    </dv>...</dataset>
</datasets>
<legend>
    <variable color="#3ca115" id="APG-DB:group-1004" selected="false">Availability (%)</
variable>
    <variable color="#ff9c00" id="APG-DB:group-2749" selected="true">LastResponseTime (ms
)</variable>
</legend>
</graph-element>
</getReportResponse>

```

```

<getReportResponse>
    <table-element id="0" name="Transactions" type="apg">
        <table>
            <header>
                <th>Transaction</th>
                <th>Server</th>
                <th>Availability (%)</th>
                <th>Avg response time (ms)</th>
                <th>Max response time (ms)</th>
                <th>Last response time (ms)</th>
            </header>
            <data>
                <tr id="31d664db">
                    <ts>ftp</ts>
                    <ts>w4n-ds-any01.int.watch4net.net</ts>
                    <tv state="ok">100</tv>
                    <tv state="major">106</tv>
                    <tv state="major">321</tv>
                    <tv state="ok">105</tv>
                </tr>...</data>
            </table>
        </table-element>
    </getReportResponse>

```

```
</table>
</table-element>
</getReportResponse>
```

2 WSDL Service Definition

This section includes the relevant *WSDL* and *XSD* files related to the *ReportManager* service definition. It can be useful to see the exact definition of an operation or a parameter using the included files.

2.1 ReportManager.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions targetNamespace="http://www.watch4net.com/APG/Remote/ReportManagerService"
  name="ReportManagerService" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://www.watch4net.com/APG/Remote/ReportManagerService"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.
  xsd"
  xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns="http://schemas.xmlsoap.org/wsdl/">

  <wsp:Policy wsu:Id="ReportManagerPortBinding_MTOM_Policy">
    <wsp:ExactlyOne>
      <wsp:All>
        <mtom:OptimizedMimeSerialization
          xmlns:mtom="http://schemas.xmlsoap.org/ws/2004/09/policy/
optimizedmimeserialization" />
        </wsp:All>
      </wsp:ExactlyOne>
    </wsp:Policy>

    <types>
      <xsd:schema>
        <xsd:import
          namespace="http://www.watch4net.com/APG/Remote/ReportManagerService"
          schemaLocation="ReportManager.xsd" />
        </xsd:schema>
      </types>

      <message name="getReport">
```

```
</part name="parameters" element="tns:getReport" />
</message>

<message name="getReportResponse">
  <part name="parameters" element="tns:getReportResponse" />
</message>

<message name="Exception">
  <part name="fault" element="tns:Exception" />
</message>

<portType name="ReportManager">
  <operation name="getReport">
    <input message="tns:getReport" />
    <output message="tns:getReportResponse" />
    <fault message="tns:Exception" name="Exception" />
  </operation>
</portType>

<binding name="ReportManagerPortBinding" type="tns:ReportManager">
  <wsp:PolicyReference URI="#ReportManagerPortBinding_MTOM_Policy"
    required="true" />
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
    style="document" />
  <operation name="getReport">
    <soap:operation soapAction="" />
    <input>
      <soap:body use="literal" />
    </input>
    <output>
      <soap:body use="literal" />
    </output>
    <fault name="Exception">
      <soap:fault name="Exception" use="literal" />
    </fault>
  </operation>
</binding>

<service name="ReportManagerService">
  <port name="ReportManagerPort" binding="tns:ReportManagerPortBinding">
    <soap:address location="http://localhost:58080/frontend-remote/wsapi/report" />
  </port>
```

```
</service>

</definitions>
```

2.2 ReportManager.xsd

```
<xs:schema version="1.0"
  targetNamespace="http://www.watch4net.com/APG/Remote/ReportManagerService"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xr="http://www.watch4net.com/APG/Web/XmlReport1"
  xmlns:xt="http://www.watch4net.com/APG/Web/XmlTree1"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  xmlns:ws="http://www.watch4net.com/APG/Remote/ReportManagerService">

  <xs:import namespace="http://www.watch4net.com/APG/Web/XmlTree1"
    schemaLocation="XmlTree1.xsd" />
  <xs:import namespace="http://www.watch4net.com/APG/Web/XmlReport1"
    schemaLocation="XmlReport1.xsd" />

  <xs:element name="getReport" type="ws:getReport" />

  <xs:element name="getReportResponse" type="ws:getReportResponse" />

  <xs:element name="Exception" type="ws:Exception" />

  <xs:complexType name="ReportProperty">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="key" type="ws:ReportPropertyKey" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name="ReportPropertyKey">
    <xs:restriction base="xs:string">
      <xs:enumeration value="apg.ws.mtom" />
      <xs:enumeration value="apg.ws.node.id" />
      <xs:enumeration value="apg.ws.report.element.with.icon" />
      <xs:enumeration
        value="apg.ws.report.element.with.preferences" />
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

```

        <xs:enumeration
            value="apg.ws.report.element.with.properties" />

        <xs:enumeration value="apg.ws.report.graph.with.image" />
        <xs:enumeration value="apg.ws.report.graph.with.data" />
        <xs:enumeration value="apg.ws.report.graph.with.legend" />
        <xs:enumeration
            value="apg.ws.report.graph.with.legend.image" />

        <xs:enumeration value="apg.ws.report.graph.width" />
        <xs:enumeration value="apg.ws.report.graph.height" />
        <xs:enumeration value="apg.ws.report.graph.scale" />
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="ReportProperties">
    <xs:sequence>
        <xs:element name="property" type="ws:ReportProperty"
            minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="getReport">
    <xs:sequence>
        <xs:element name="properties" type="ws:ReportProperties"
            minOccurs="0" maxOccurs="1" />
        <xs:element ref="xt:node" />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="getReportResponse">
    <xs:sequence>
        <xs:group ref="xr:ReportElementChoice" />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="Exception">
    <xs:sequence>
        <xs:element name="message" type="xs:string" minOccurs="0" />
    </xs:sequence>
</xs:complexType>

```

```
</xs:schema>
```

2.3 XmlTree1.xsd

```
<?xml version="1.0" encoding="UTF-8"?><xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xt="http://www.watch4net.com/APG/Web/XmlTree1" targetNamespace="http://www.watch4net.com/APG/Web/XmlTree1" attributeFormDefault="unqualified" elementFormDefault="qualified">

  <!-- ***** -->
  <!-- * THE APG XML TREE ROOT ELEMENT * -->
  <!-- ***** -->

  <!-- $Id: XmlTree1.xsd 50389 2014-07-09 19:36:29Z jbsilvy $ -->
  <xs:element name="node" type="xt:RealNode"></xs:element>

  <!-- ***** -->
  <!-- * THE NODES * -->
  <!-- ***** -->

  <xs:complexType abstract="true" name="DynamicNode">
    <xs:attribute name="singleNodeId" use="optional" type="xt:singleNodeId"></xs:attribute>
    <xs:attribute name="uid" use="optional" type="xt:nodeUid"></xs:attribute>
    <xs:attribute name="order" use="optional" type="xs:int"></xs:attribute>
  </xs:complexType>

  <xs:complexType name="RealNode">
    <xs:complexContent>
      <xs:extension base="xt:DynamicNode">
        <xs:sequence maxOccurs="1" minOccurs="1">
          <xs:element maxOccurs="unbounded" name="property" type="xt:NodeProperty"
minOccurs="0"></xs:element>
          <xs:group ref="xt:ExternalNodeProperties" maxOccurs="unbounded" minOccurs="0"
></xs:group>
          <xs:choice maxOccurs="unbounded" minOccurs="0">
            <xs:element name="link" type="xt:LinkedNode"></xs:element>
            <xs:element name="node" type="xt:RealNode"></xs:element>
          </xs:choice>
        </xs:sequence>
        <xs:attribute name="name" use="required" type="xs:string"></xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

</xs:schema>
```

```

        <xs:attribute name="type" use="optional" type="xs:string"/></xs:attribute>
    </xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="LinkedNode">
    <xs:complexContent>
        <xs:extension base="xt:DynamicNode">
            <xs:attribute name="linkID" use="required" type="xs:string"/></xs:attribute>
            <xs:attribute default="false" name="hiddenIfBroken" use="optional" type="
xs:boolean"/></xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<!-- ***** -->
<!-- * THE EXTERNAL NODE PROPERTIES * -->
<!-- ***** -->

<xs:group name="ExternalNodeProperties">
    <xs:choice><xs:element name="formula" type="xt:Formula"/></xs:element><xs:element name="
heatmap-report" type="xt:HeatmapReport"/></xs:element><xs:element name="threshold" type="
xt:Threshold"/></xs:element><xs:element name="axis" type="xt:Axis"/></xs:element><xs:element name
="customization" type="xt:MetricCustomization"/></xs:element><xs:element name="topn-report" type
="xt:TopNReport"/></xs:element><xs:element name="status-report" type="xt:StatusReport"/></
xs:element><xs:element name="treemap-report" type="xt:TreemapReport"/></xs:element><xs:element
name="wiring" type="xt:Wiring"/></xs:element></xs:choice>
</xs:group>

<!-- ***** -->
<!-- * THE NODE PROPERTIES * -->
<!-- ***** -->

<xs:complexType abstract="true" name="NodeProperty"/></xs:complexType>

<xs:complexType name="NodeFilter">
    <xs:complexContent>
        <xs:extension base="xt:NodeProperty">
            <xs:attribute default="" name="filterExpression" use="optional" type="xs:string"
>

            <xs:annotation>
                <xs:documentation>

```

```

        The filter expression.
    </xs:documentation>
</xs:annotation>
</xs:attribute>
<xs:attribute default="false" name="selectUnmatched" use="optional" type="
xt:selectUnmatchedMode">
    <xs:annotation>
        <xs:documentation>
            Specify if the node will use it's own filter or create a runtime
filter that will select the
            unmatched values from either the edit mode or the browse mode.
        </xs:documentation>
    </xs:annotation>
</xs:attribute>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="NodeExpansion">
    <xs:complexContent>
        <xs:extension base="xt:NodeProperty">
            <xs:attribute name="expandOn" use="required" type="xs:string">
                <xs:annotation>
                    <xs:documentation>
                        The expansion string, which is a comma-separated list of either
property names or complex
                        expansions.
                    </xs:documentation>
                </xs:annotation>
            </xs:attribute>
            <xs:attribute default="expand" name="filterMode" type="xt:filterMode">
                <xs:annotation>
                    <xs:documentation>
                        This value sets the way the filter is applied on multi-nodes.
                    </xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType abstract="true" name="DynamicSetting">

```

```

        <xs:complexContent>
          <xs:extension base="xt:NodeProperty"></xs:extension>
        </xs:complexContent>
      </xs:complexType>

      <xs:complexType name="PropertyReplaceSetting">
        <xs:complexContent>
          <xs:extension base="xt:DynamicSetting">
            <xs:attribute name="pattern" use="required" type="xs:string"></xs:attribute>
            <xs:attribute name="target" use="required" type="xs:string"></xs:attribute>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>

      <xs:complexType name="DisplayPreferences">
        <xs:complexContent>
          <xs:extension base="xt:NodeProperty">
            <xs:attribute default="neverHide" name="displayPolicy" use="optional" type="
xt:displayPolicy"></xs:attribute>
            <xs:attribute default="false" name="browsingDisabled" use="optional" type="
xs:boolean"></xs:attribute>
            <xs:attribute default="false" name="propagateTimeSelection" use="optional" type=
"xs:boolean"></xs:attribute>
            <xs:attribute default="d" name="legendVisibility" use="optional" type="xs:string
"></xs:attribute>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>

      <xs:complexType name="ReportPreferences">
        <xs:complexContent>
          <xs:extension base="xt:NodeProperty">
            <xs:attribute default="0" name="displayMode" use="optional" type="
xt:reportDisplayMode"></xs:attribute>
            <xs:attribute default="srt" name="defaultMode" use="optional" type="
xt:reportMode"></xs:attribute>
            <xs:attribute default="2" name="duration" use="optional" type="xt:duration"></
xs:attribute>
            <xs:attribute name="timeZoneId" use="optional" type="xs:string"></xs:attribute>
            <xs:attribute default="0" name="preferredPeriod" use="optional" type="
xt:aggregationPeriod"></xs:attribute>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>

```

```

        <xs:attribute default="average" name="preferredAggregate" use="optional" type="
xt:aggregationFunction"></xs:attribute>
        <xs:attribute name="timeFilterExpression" use="optional" type="
xt:timeFilterExpression"></xs:attribute>
        <xs:attribute name="displayedProperties" use="optional" type="xs:string"></
xs:attribute>
        <xs:attribute name="description" use="optional" type="xs:string"></xs:attribute>
        <xs:attribute default="name_unit" name="legendProperties" use="optional" type="
xs:string"></xs:attribute>
        <xs:attribute default="d" name="legendDisplayPolicy" use="optional" type="
xs:string"></xs:attribute>
        <xs:attribute default="true" name="displayUnselectedVariables" use="optional"
type="xs:boolean"></xs:attribute>
        <xs:attribute name="criticalThreshold" use="optional" type="xs:float"></
xs:attribute>
        <xs:attribute name="majorThreshold" use="optional" type="xs:float"></
xs:attribute>
        <xs:attribute name="maxValue" use="optional" type="xs:float"></xs:attribute>
        <xs:attribute default="0" name="minValue" use="optional" type="xs:float"></
xs:attribute>
        <xs:attribute default="1" name="graphScaleFactor" use="optional" type="xs:string
"></xs:attribute>
        <xs:attribute default="none" name="graphInfoDisplayMode" use="optional" type="
xt:graphInfoDisplayMode"></xs:attribute>
        <xs:attribute default="25" name="paging" use="optional" type="xt:paging"></
xs:attribute>
        <xs:attribute default="5" name="treePaging" use="optional">
        <xs:simpleType>
            <xs:restriction base="xs:int">
                <xs:minExclusive value="0"></xs:minExclusive>
            </xs:restriction>
        </xs:simpleType>
        </xs:attribute>
        <xs:attribute default="1" name="propertyLimit" use="optional" type="xs:int"></
xs:attribute>
        <xs:attribute default="240" name="dataLifeTime" use="optional" type="xs:int"></
xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="RuntimePreferences">

```

```

        <xs:complexContent>
          <xs:extension base="xt:NodeProperty">
            <xs:attribute name="timeRangeExpression" use="optional" type="xt:timeRange"></
xs:attribute>
            <xs:attribute name="selectedVariablesString" use="optional" type="
xt:selectedVariableList"></xs:attribute>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>

      <!-- ***** -->
      <!-- * THE NODE COLUMNS * -->
      <!-- ***** -->

      <xs:simpleType name="periodEnforcement">
        <xs:restriction base="xs:string">
          <xs:enumeration value="true"></xs:enumeration>
          <xs:enumeration value="false"></xs:enumeration>
          <xs:enumeration value="never"></xs:enumeration>
        </xs:restriction>
      </xs:simpleType>

      <xs:complexType abstract="true" name="NodeColumn">
        <xs:complexContent>
          <xs:extension base="xt:NodeProperty">
            <xs:attribute name="name" use="required" type="xs:string"></xs:attribute>
            <xs:attribute default="none" name="sortMode" use="optional" type="xt:sortMode"><
/xs:attribute>
            <xs:attribute default=".*" name="columnFilterCondition" use="optional" type="
xs:string"></xs:attribute>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>

      <xs:complexType abstract="true" name="VariableNodeColumn">
        <xs:complexContent>
          <xs:extension base="xt:NodeColumn">
            <xs:attribute name="resultName" use="optional" type="xs:string"></xs:attribute>
            <xs:attribute default="" name="filterExpression" use="optional" type="xs:string"
></xs:attribute>
            <xs:attribute default="inherit" name="period" use="optional" type="
xt:inheritedAggregationPeriod"></xs:attribute>

```

```

        <xs:attribute default="inherit" name="aggregationFunc" use="optional" type="
xt:inheritedAggregationFunction"></xs:attribute>
        <xs:attribute name="durationOverride" use="optional" type="xt:timeRange"></
xs:attribute>
        <xs:attribute name="durationPartition" use="optional" type="xt:durationValue"></
xs:attribute>
        <xs:attribute default="0" name="timeThreshold" use="optional">
        <xs:simpleType>
        <xs:restriction base="xs:int">
        <xs:minInclusive value="0"></xs:minInclusive>
        </xs:restriction>
        </xs:simpleType>
        </xs:attribute>
        <xs:attribute default="inherit" name="valuesAggregationFunc" use="optional" type
="xt:inheritedAggregationFunction"></xs:attribute>
        <xs:attribute default="false" name="perLineTimeFilter" use="optional" type="
xs:boolean"></xs:attribute>
        </xs:extension>
        </xs:complexContent>
        </xs:complexType>

        <xs:complexType name="PropertyNodeColumn">
        <xs:complexContent>
        <xs:extension base="xt:VariableNodeColumn">
        <xs:attribute name="property" use="required" type="xs:string"></xs:attribute>
        <xs:attribute default="25" name="limit" use="optional" type="xs:int"></
xs:attribute>
        <xs:attribute default="true" name="useTimeRange" use="optional" type="xs:boolean
"></xs:attribute>
        <xs:attribute default="never" name="forcePeriod" use="optional" type="
xt:periodEnforcement"></xs:attribute>
        </xs:extension>
        </xs:complexContent>
        </xs:complexType>

        <xs:complexType name="NodePropertyNodeColumn">
        <xs:complexContent>
        <xs:extension base="xt:NodeColumn">
        <xs:attribute name="nodeProperty" use="required" type="xs:string"></xs:attribute
>
        </xs:extension>
        </xs:complexContent>

```

```

</xs:complexType>

<xs:complexType name="ValueNodeColumn">
  <xs:complexContent>
    <xs:extension base="xt:VariableNodeColumn">
      <xs:attribute default="false" name="useTimeRange" use="optional" type="
xs:boolean"/></xs:attribute>
      <xs:attribute default="false" name="forcePeriod" use="optional" type="
xt:periodEnforcement"/></xs:attribute>
      <xs:attribute name="majorLevel" use="optional" type="xs:double"/></xs:attribute>
      <xs:attribute default="false" name="nonMajorHidden" use="optional" type="
xs:boolean"/></xs:attribute>
      <xs:attribute name="criticalLevel" use="optional" type="xs:double"/></
xs:attribute>
      <xs:attribute name="scaledLevels" use="optional" type="xs:boolean"/></
xs:attribute>
      <xs:attribute default="false" name="nonCriticalHidden" use="optional" type="
xs:boolean"/></xs:attribute>
      <xs:attribute default="false" name="useComplement" use="optional" type="
xs:boolean"/></xs:attribute>
      <xs:attribute default="false" name="replaceNullWithZero" use="optional" type="
xs:boolean"/></xs:attribute>
      <xs:attribute default="GraphFormatter" name="formatter" use="optional" type="
xt:valueFormatter"/></xs:attribute>
      <xs:attribute name="criticityAsc" use="optional" type="xs:boolean"/></
xs:attribute>
      <xs:attribute default="2" name="roundingAccuracy" use="optional">
        <xs:simpleType>
          <xs:restriction base="xs:int">
            <xs:minInclusive value="-1"/></xs:minInclusive>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute default="none" name="scaleOperation" use="optional" type="
xt:scaleOperation"/></xs:attribute>
      <xs:attribute default="1" name="scaleFactor" use="optional" type="xs:string"/></
xs:attribute>
      <xs:attribute name="displayWhen" use="optional" type="xs:string"/></xs:attribute>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

```

<!-- ***** -->
<!-- * SIMPLE TYPES * -->
<!-- ***** -->

<xs:simpleType name="singleNodeId">
  <xs:annotation>
    <xs:documentation>
      A single node id is the unique identifier of a node. It must be an hexadecimal
notation of a
      number.
      This
      id must me unique among the node's siblings. The globally unique node identifier
is the concatenation of
      its
      parent single
      node ids and its own single node id (using '-' as a separator).
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="[a-f0-9]+"></xs:pattern>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="nodeUid">
  <xs:annotation>
    <xs:documentation>
      A node uid is an id which is supposed to be globally unique among the tree,
contrary to the a
      single
      node
      id, which is only unique among the node's siblings. This is intended to be used
with links, since it is
      stronger that
      single node ids, and user generally has control over it.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="[a-zA-Z0-9\.\-_]+"></xs:pattern>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="inheritedValue">

```

```

        <xs:restriction base="xs:string">
            <xs:enumeration value="inherit"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="aggregationPeriod">
        <xs:restriction base="xs:int">
            <xs:minInclusive value="0"></xs:minInclusive>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="inheritedAggregationPeriod">
        <xs:union memberTypes="xt:aggregationPeriod xt:inheritedValue"></xs:union>
    </xs:simpleType>

    <xs:simpleType name="aggregationFunction">
        <xs:annotation>
            <xs:documentation>
                This is the APG built-in aggregation functions. These functions are natively
supported by APG and
                they
                usually map to some pre-computed aggregates in database. Some functions (last,
lasttimestamp) are meaningless
                with
                spacial
                aggregations, and then may not be honored, depending on the report.
            </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="average"></xs:enumeration>
            <xs:enumeration value="min"></xs:enumeration>
            <xs:enumeration value="max"></xs:enumeration>
            <xs:enumeration value="sum"></xs:enumeration>
            <xs:enumeration value="last"></xs:enumeration>
            <xs:enumeration value="nbvalues"></xs:enumeration>
            <xs:enumeration value="lasttimestamp"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="inheritedAggregationFunction">
        <xs:union memberTypes="xt:aggregationFunction xt:inheritedValue"></xs:union>
    </xs:simpleType>

```

```

<xs:simpleType name="timeRange">
  <xs:annotation>
    <xs:documentation>
      A time range can be either a relative duration (last day, previous day, ...) or
a specific window
      (with a start time and a end time). The time drift is not supported anymore even
if the syntax is still supported.
      If the same fonctionnality is required, use timestamps.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="d:(-102|-101|-100|-99|-98|-97|-96|-95)(-?[0-9]+)?">
      <xs:annotation>
        <xs:documentation>
          Deprecated way of giving the time range in a relative duration. It will
select the previous day,
          week,
          ... relative to now. See duration type description for details. The last
argument is the time drift.
        </xs:documentation>
      </xs:annotation>
    </xs:pattern>
    <xs:pattern value="d:(-2|-1|0|1|2|3|4|5)(-?[0-9]+)?">
      <xs:annotation>
        <xs:documentation>
          Deprecated way of giving the time range in a relative duration. It will
select the last day,
          week,
          ...
          relative to now. See duration type description for details. The last
argument is the time drift.
        </xs:documentation>
      </xs:annotation>
    </xs:pattern>
    <xs:pattern value="d:(98|99|100|101|102|103|104|105)(-?[0-9]+)?">
      <xs:annotation>
        <xs:documentation>
          Deprecated way of giving the time range in a relative duration. It will
select the current day,
          week,

```

```

... relative to now. See duration type description for details. The last
argument is the time drift.
</xs:documentation>
</xs:annotation>
</xs:pattern>
<xs:pattern value="r:[0-9]+:[0-9]+(:-?[0-9]+)?">
  <xs:annotation>
    <xs:documentation>
      This time range is a specific window. The range is specified giving two
UNIX timestamps (number
of
seconds since epoch). The last argument is the time drift.
    </xs:documentation>
  </xs:annotation>
</xs:pattern>
<xs:pattern value="d:([plc](\d+[smhdwMy])+)(:-?[0-9]+)?">
  <xs:annotation>
    <xs:documentation>
      The recommended way of giving the time range in a relative duration. The
first argument is the
relative
to option. The "p" option stands for previous while the "l" option stands
for last and the "c" option
stands for
current. The next part is made of numbers and units to represent the
duration.

      The units are:
      s = second
      m = minute
      h =
      hour
      d = day
      w = week
      M = month
      y = year

      There's no limit on the number of units, but each unit can only
      be present once.
      For
      example, 1h45m will be translated as 1 hour and 45 minutes. The last
argument is the time

```

```

        drift.
      </xs:documentation>
    </xs:annotation>
  </xs:pattern>
  <xs:pattern value="(-102|-101|-100|-99|-98|-97|-96|-95)(:~?[0-9]+)?">
    <xs:annotation>
      <xs:documentation>
        Deprecated way of giving the time range. Identical to pattern "d:
        (-102|-101|-100|-99|-98|-97|-96|-95)(:~?[0-9]+)?" .
      </xs:documentation>
    </xs:annotation>
  </xs:pattern>
  <xs:pattern value="(-2|-1|0|1|2|3|4|5)(:~?[0-9]+)?">
    <xs:annotation>
      <xs:documentation>
        Deprecated way of giving the time range. Identical to pattern "d:
        (-2|-1|0|1|2|3|4|5)(:~?[0-9]+)?" .
      </xs:documentation>
    </xs:annotation>
  </xs:pattern>
  <xs:pattern value="(98|99|100|101|102|103|104|105)(:~?[0-9]+)?">
    <xs:annotation>
      <xs:documentation>
        Deprecated way of giving the time range. Identical to pattern "d:
        (98|99|100|101|102|103|104|105)(:~?[0-9]+)?" .
      </xs:documentation>
    </xs:annotation>
  </xs:pattern>
  <xs:pattern value="([plc](\d+[smhdwMy]))(:~?[0-9]+)?">
    <xs:annotation>
      <xs:documentation>
        Deprecated way of giving the time range. Identical to pattern "d:([plc](\
d+[smhdwMy]))(:~?[0-9]+)?" .
      </xs:documentation>
    </xs:annotation>
  </xs:pattern>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="selectedVariableList">
  <xs:annotation>
    <xs:documentation>

```

```

        A variable list looks like var1_var2_... This list is used to exclude variable
from a specific
    report.
    Variable name format depends on the selected report mode.
</xs:documentation>
</xs:annotation>
<xs:restriction base="xs:string">
    <xs:pattern value="([_]+(_[_]+)*)?"></xs:pattern>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="durationValue">
    <xs:annotation>
        <xs:documentation>
            A duration value is the length of a duration.
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
        <xs:pattern value="(\d+[smhdwMy])+>
            <xs:annotation>
                <xs:documentation>
                    The duration value is made of numbers and units to represent the duration
                    .

                    The units are:
                    s = second
                    m = minute
                    h =
                    hour
                    d = day
                    w = week
                    M = month
                    y = year

                    There's no limit on the number of units, but each unit can only
                    be present once.
                    For
                    example, 1h45m will be translated as 1 hour and 45 minutes.
                </xs:documentation>
            </xs:annotation>
        </xs:pattern>
    </xs:restriction>

```

```

</xs:simpleType>

<xs:simpleType name="duration">
  <xs:annotation>
    <xs:documentation>
      A duration is a code which designate the APG periods.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="(-102|-101|-100|-99|-98|-97|-96|-95)">
      <xs:annotation>
        <xs:documentation>
          Deprecated way of giving the time range in a relative duration. It will
select the previous
          day,
          week,
          ... relative to the current time. See duration type description for
details.
        </xs:documentation>
      </xs:annotation>
    </xs:pattern>
    <xs:pattern value="(-2|-1|0|1|2|3|4|5)">
      <xs:annotation>
        <xs:documentation>
          Deprecated way of giving the time range in a relative duration. It will
select the last
          day, week,
          ...
          relative to the current time. See duration type description for details.
        </xs:documentation>
      </xs:annotation>
    </xs:pattern>
    <xs:pattern value="(98|99|100|101|102|103|104|105)">
      <xs:annotation>
        <xs:documentation>
          Deprecated way of giving the time range in a relative duration. It will
select the current
          day,
          week,
          ... relative to the current time. See duration type description for
details.
        </xs:documentation>
      </xs:annotation>
    </xs:pattern>
  </xs:restriction>

```

```

        </xs:annotation>
    </xs:pattern>
    <xs:pattern value="[plc](\d+[smhdwMy])+>
        <xs:annotation>
            <xs:documentation>
                The recommended way of giving the time range in a relative duration. The
first argument is the
                relative
                to option. The "p" option stands for previous while the "l" option stands
for last and the "c" option
                stands for
                current. The next part is made of numbers and units to represent the
duration.

                The units are:
                s = second
                m = minute
                h =
                hour
                d = day
                w = week
                M = month
                y = year

                There's no limit on the number of units, but each unit can only
                be present once.
                For
                example, 1h45m will be translated as 1 hour and 45 minutes.
            </xs:documentation>
        </xs:annotation>
    </xs:pattern>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="reportMode">
    <xs:annotation>
        <xs:documentation>
            The report mode control how the report is generated. The default modes are the
reports that are
            bundeled
            with

```

```

        every frontend based application. The external report modes usually rely on
        plugins, and might not be
        active.
    </xs:documentation>
</xs:annotation>
<xs:union memberTypes="xt:defaultReportMode_␣xt:externalReportMode"></xs:union>
</xs:simpleType>

<xs:simpleType name="defaultReportMode">
    <xs:annotation>
        <xs:documentation>
            The default report modes.
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
        <xs:enumeration value="srt">
            <xs:annotation>
                <xs:documentation>
                    The table mode is a report mode which displays one row per child node. If
                    the selected node is a
                    leaf
                    (no child), it will display a table using some pseudo children, which
                    depend on the database scheme.
                    For the
                    APG database
                    schema, you will see one line per indicator.
                </xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="lrt">
            <xs:annotation>
                <xs:documentation>
                    The leaf table mode. It works like the table mode but its behavior is
                    always the same as if the
                    selected node was a leaf.
                </xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="nrx">
            <xs:annotation>
                <xs:documentation>

```

```

        The node report mode stands for the graph report that gather all the
variables from the currently
        selected node. The report is built drawing a single graph that contains
all the variables.
        </xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="nrb">
        <xs:annotation>
            <xs:documentation>
                This report mode shows a single graph report that displays the metrics
available on this node in
                vertical bars.
            </xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="ncr">
        <xs:annotation>
            <xs:documentation>
                The node child report mode works like the stacked chart mode but instead
of plotting series on an
                area
                chart, they are plotted using simple lines.
            </xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="srg">
        <xs:annotation>
            <xs:documentation>
                The node child report mode works like the node mode except that actual
data is taken from child
                nodes
                instead of current node. Using this mode, you gets a single graph with
one line per child node.
            </xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="frg">
        <xs:annotation>
            <xs:documentation>
                The full graph report mode mixes one node report per leaf of the tree.
You gets a separate graph
            </xs:documentation>
        </xs:annotation>
    </xs:enumeration>

```

```
for
    each leaf of the sub-tree. Graphs are organized in a hierarchical way
that follows the organization of the
    tree.
</xs:documentation>
</xs:annotation>
</xs:enumeration>
<xs:enumeration value="vrz">
    <xs:annotation>
        <xs:documentation>
            The variable report mode stands for the base line report mode. This
report shows a single metric
            baseline, and its variations among time.
        </xs:documentation>
    </xs:annotation>
</xs:enumeration>
<xs:enumeration value="stk">
    <xs:annotation>
        <xs:documentation>
            The stacked chart mode displays one serie per child on a stacked area
chart.
        </xs:documentation>
    </xs:annotation>
</xs:enumeration>
<xs:enumeration value="stb">
    <xs:annotation>
        <xs:documentation>
            This report mode shows a single graph report where each metric is stacked
, one over the other, in
            vertical bars.
        </xs:documentation>
    </xs:annotation>
</xs:enumeration>
<xs:enumeration value="hb">
    <xs:annotation>
        <xs:documentation>
            The horizontal bar report mode is a mode which aggregates data for each
children using the
            specified
            aggregation function over the selected period. The display is in
percentage.
        </xs:documentation>
    </xs:annotation>
</xs:enumeration>
```

```

        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="pie">
        <xs:annotation>
            <xs:documentation>
                The pie chart report mode works like the horizontal bar mode but displays
the result as a pie.
            </xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="gg">
        <xs:annotation>
            <xs:documentation>
                The gauge chart report mode displays a single value at a time, which is
the aggregation of the
                selected child using the specified aggregation function over the selected
period.
            </xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="ico">
        <xs:annotation>
            <xs:documentation>
                The icon mode shows one icon per child. It doesn't use any performance
data and is only useful
                for
                nice
                report selection panel.
            </xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="mix">
        <xs:annotation>
            <xs:documentation>
                The mixed report mode is used to combine several of the previously
mentioned modes. This mode
                uses
                the
                report mode from any child node but keeps the selected period and
aggregation from the selected node.
            </xs:documentation>
        </xs:annotation>
    </xs:enumeration>

```

```

        </xs:enumeration>
        <xs:enumeration value="dmx">
          <xs:annotation>
            <xs:documentation>
              The mixed defaults report mode works like the mixed report mode but uses
the default report
              settings
              from any child node.
            </xs:documentation>
          </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="rmx">
          <xs:annotation>
            <xs:documentation>
              The mixed runtime report mode works like the mixed report mode but uses
the runtime report
              settings
              from any child node.
            </xs:documentation>
          </xs:annotation>
        </xs:enumeration>
      </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="externalReportMode">
      <xs:annotation>
        <xs:documentation>
          The external report modes, generally handled by plugins.
        </xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:string"></xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="reportDisplayMode">
      <xs:annotation>
        <xs:documentation>
          The report display mode control how to display the report elements. This may not
be used by every
          renderers.
        </xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:int">

```

```

        <xs:enumeration value="0">
          <xs:annotation>
            <xs:documentation>
              The default rendering mode: one element per line, with informations.
            </xs:documentation>
          </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="1">
          <xs:annotation>
            <xs:documentation>
              Displays one element per line stripping out properties and legend for
graphs.
            </xs:documentation>
          </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="2">
          <xs:annotation>
            <xs:documentation>
              Displays two elements per line stripping out properties and legend for
graphs.
            </xs:documentation>
          </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="3">
          <xs:annotation>
            <xs:documentation>
              Displays three elements per line stripping out properties and legend for
graphs.
            </xs:documentation>
          </xs:annotation>
        </xs:enumeration>
      </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="timeFilterExpression">
      <xs:annotation>
        <xs:documentation>
          A time filter expression describes a time filter. There are several types of
time filters.
        </xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:string">

```

```

        <xs:pattern value="simple:(([1-9]|[1-2][0-9]|3[0-1])((,[1-9]|[1-2][0-9]|3[0-1]))*)
        ?;(([1-7])((,[1-7]))*)?;([0-9]|1[0-9]|2[0-3])((,[0-9]|1[0-9]|2[0-3]))*)?">
        <xs:annotation>
            <xs:documentation>
                A simple time filter will exclude periods of time based on a specific
                occurrence. There are three
                arguments, respectively the day of month (between 1 and 31), the day of
                week (between 1 and 7) and the hour of day
                (between 0 and 23) to exclude.
            </xs:documentation>
        </xs:annotation>
    </xs:pattern>
    <xs:pattern value="remote:[^:]+\..*">
        <xs:annotation>
            <xs:documentation>
                A remote time filter will used outages from a remote outage manager. The
                first argument is the
                outage
                manager alias, which is defined in engine configuration and the second
                argument is the remote object name.
            </xs:documentation>
        </xs:annotation>
    </xs:pattern>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="valueFormatter">
    <xs:annotation>
        <xs:documentation>
            A value formatter is used to change the ways a table cell value is displayed.
            Actual effect depends
            on
            the way the report is rendered.
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
        <xs:pattern value="BlankFormatter">
            <xs:annotation>
                <xs:documentation>
                    Hides the value.
                </xs:documentation>
            </xs:annotation>
        </xs:pattern>
    </xs:restriction>

```

```
</xs:pattern>
<xs:pattern value="BasicFormatter">
  <xs:annotation>
    <xs:documentation>
      Simply displays the value, without any formatting.
    </xs:documentation>
  </xs:annotation>
</xs:pattern>
<xs:pattern value="BackgroundFormatter">
  <xs:annotation>
    <xs:documentation>
      Displays the value and change the background of the cell according to the
thresholds.
    </xs:documentation>
  </xs:annotation>
</xs:pattern>
<xs:pattern value="GraphFormatter">
  <xs:annotation>
    <xs:documentation>
      This formatter is used for HTML based report. The value should be
decorated with a visual
      threshold
      and
      clicking on it should trigger an instant history. Most renderers will not
use it.
    </xs:documentation>
  </xs:annotation>
</xs:pattern>
<xs:pattern value="DateTimeFormatter">
  <xs:annotation>
    <xs:documentation>
      Formats the value as a date with time.
    </xs:documentation>
  </xs:annotation>
</xs:pattern>
<xs:pattern value="TimeFormatter">
  <xs:annotation>
    <xs:documentation>
      Formats the value as a date.
    </xs:documentation>
  </xs:annotation>
</xs:pattern>
```

```

<xs:pattern value="DurationFormatter">
  <xs:annotation>
    <xs:documentation>
      Formats the value as a duration.
    </xs:documentation>
  </xs:annotation>
</xs:pattern>
<xs:pattern value="SparkLineFormatter">
  <xs:annotation>
    <xs:documentation>
      Shows the value along with a quick overview of its history over the
selected period.
    </xs:documentation>
  </xs:annotation>
</xs:pattern>
<xs:pattern value="StatusSparkLineFormatter(&lt;.*&gt;)?">
  <xs:annotation>
    <xs:documentation>
      Shows the value as a performance relative to the other values on the same
page.
    </xs:documentation>
  </xs:annotation>
</xs:pattern>
<xs:pattern value="PerformanceSparkLineFormatter">
  <xs:annotation>
    <xs:documentation>
      Shows the value along with its status over the selected period.
    </xs:documentation>
  </xs:annotation>
</xs:pattern>
<xs:pattern value="BarChartSparkLineFormatter(&lt;.*&gt;)?">
  <xs:annotation>
    <xs:documentation>
      Shows the value along with its history and its status over the selected
period.
    </xs:documentation>
  </xs:annotation>
</xs:pattern>
<xs:pattern value="ValueToStringFormatter(&lt;.*&gt;)?">
  <xs:annotation>
    <xs:documentation>
      Converts a given value into a string

```

```

        </xs:documentation>
      </xs:annotation>
    </xs:pattern>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="scaleOperation">
  <xs:annotation>
    <xs:documentation>
      A scale operation is either a multiplication or a division on a value. It is
used along with a
      scale
      factor.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="none">
      <xs:annotation>
        <xs:documentation>
          Doesn't scale the value.
        </xs:documentation>
      </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="multiply">
      <xs:annotation>
        <xs:documentation>
          Multiplies the value with the scale factor.
        </xs:documentation>
      </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="divide">
      <xs:annotation>
        <xs:documentation>
          Divides the value by the scale factor.
        </xs:documentation>
      </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="unit">
      <xs:annotation>
        <xs:documentation>
          Scales the value according to its unit.
        </xs:documentation>
      </xs:annotation>
    </xs:enumeration>
  </xs:restriction>
</xs:simpleType>

```

```

        </xs:annotation>
      </xs:enumeration>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="displayPolicy">
    <xs:annotation>
      <xs:documentation>
        The display policy controls wether a specific node should be visible or not.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="neverHide">
        <xs:annotation>
          <xs:documentation>
            The node will never be hidden.
          </xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="hideIfEmpty">
        <xs:annotation>
          <xs:documentation>
            The node will be hidden if its filter doesn't select anything.
          </xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="hideIfLeaf">
        <xs:annotation>
          <xs:documentation>
            The node will be hidden if it is a leaf. The test will be achieved after
node expansion. If there
            is
            no
            expanded children, the node will be considered a leaf and then will be
hidden.
          </xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="hideIfLeafOrEmpty">
        <xs:annotation>
          <xs:documentation>
            The node will be hidden if it's either a leaf or doesn't select anything.

```

```

        </xs:documentation>
      </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="hideIfBrowsing">
      <xs:annotation>
        <xs:documentation>
          This node will be hidden in the tree in Browse mode but will be used to
compute report.
        </xs:documentation>
      </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="hideInTreeAndReport">
      <xs:annotation>
        <xs:documentation>
          This node will be hidden in the tree and reports in Browse mode but used
for formula results.
        </xs:documentation>
      </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="alwaysHide">
      <xs:annotation>
        <xs:documentation>
          The node will always be hidden, useful for templates library.
        </xs:documentation>
      </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="hideIfNotLinked">
      <xs:annotation>
        <xs:documentation>
          The node will always be hidden, except if it's accessed through a link.
        </xs:documentation>
      </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="hideInReport">
      <xs:annotation>
        <xs:documentation>
          The node will always be hidden in report but not in the tree.
        </xs:documentation>
      </xs:annotation>
    </xs:enumeration>
  </xs:restriction>
</xs:simpleType>

```

```
<xs:simpleType name="sortMode">
  <xs:annotation>
    <xs:documentation>
      The sorting mode of a table column.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="asc(#[0-9]+)?">
      <xs:annotation>
        <xs:documentation>
          Sorts the column using ascending order. The optional sort order (after
the #) allows to specify a
          multi column sort in a different order that the columns.
        </xs:documentation>
      </xs:annotation>
    </xs:pattern>
    <xs:pattern value="desc(#[0-9]+)?">
      <xs:annotation>
        <xs:documentation>
          Sorts the column using descending order. The optional sort order (after
the #) allows to specify
          a
          multi column sort in a different order that the columns.
        </xs:documentation>
      </xs:annotation>
    </xs:pattern>
    <xs:pattern value="none">
      <xs:annotation>
        <xs:documentation>
          Doesn't sort the column.
        </xs:documentation>
      </xs:annotation>
    </xs:pattern>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="filterMode">
  <xs:annotation>
    <xs:documentation>
      The filter mode control how a filter affects the node expansion.
    </xs:documentation>
  </xs:annotation>
```

```

</xs:annotation>
<xs:restriction base="xs:string">
  <xs:enumeration value="expand">
    <xs:annotation>
      <xs:documentation>
        The filter will be used to restrict expansions but will not affect
selection.
      </xs:documentation>
    </xs:annotation>
  </xs:enumeration>
  <xs:enumeration value="select">
    <xs:annotation>
      <xs:documentation>
        The filter will not be used to restrict expansions but will restrict
selection.
      </xs:documentation>
    </xs:annotation>
  </xs:enumeration>
  <xs:enumeration value="both">
    <xs:annotation>
      <xs:documentation>
        The filter will be used to restrict expansions and selection.
      </xs:documentation>
    </xs:annotation>
  </xs:enumeration>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="selectUnmatchedMode">
  <xs:annotation>
    <xs:documentation>
      This specify if the node will use either it's normal filter (false), a filter
that will select the
      unmatched values of the
      same level nodes from the edit mode filter (true) or a filter that will select
the unmatched
      values of the same level
      nodes from the browse mode filter (browse)
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="true"></xs:enumeration>

```

```

        <xs:enumeration value="false"></xs:enumeration>
        <xs:enumeration value="browse"></xs:enumeration>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="graphInfoDisplayMode">
    <xs:annotation>
        <xs:documentation>
            The graph info display mode control how additional information will be displayed
on a graph.
        </xs:documentation>
    </xs:annotation>
    <xs:list>
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="none">
                    <xs:annotation>
                        <xs:documentation>
                            This mode doesn't show any information on the graph
                        </xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="pinpoint">
                    <xs:annotation>
                        <xs:documentation>
                            This mode will "point" some information at several places on the
graph.
                        </xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="boxed">
                    <xs:annotation>
                        <xs:documentation>
                            This mode will show all the information in a single box within the
graph.
                        </xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="trend">
                    <xs:annotation>
                        <xs:documentation>
                            This mode will show the trending curve of the current data.

```

```

        </xs:documentation>
      </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="logarithmic">
      <xs:annotation>
        <xs:documentation>
          This mode will show the graph using a logarithmic range axis.
        </xs:documentation>
      </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="composite">
      <xs:annotation>
        <xs:documentation>
          Displays small graphs for different time ranges.
        </xs:documentation>
      </xs:annotation>
    </xs:enumeration>
  </xs:restriction>
</xs:simpleType>
</xs:list>
</xs:simpleType>

<xs:simpleType name="paging">
  <xs:annotation>
    <xs:documentation>
      The table paging, which determines how table reports are displayed. This is
      composed of two parts:
      a
      prefix (!,-,empty) and an integer value.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9]+">
      <xs:annotation>
        <xs:documentation>
          This paging mode will display all values using pages of the given size.
          If the value is 0, all
          values
          will be displayed in a single page.
        </xs:documentation>
      </xs:annotation>
    </xs:pattern>
  </xs:restriction>
</xs:simpleType>

```

```

        <xs:pattern value="-[0-9]+">
        <xs:annotation>
        <xs:documentation>
            This paging mode will display the first values of the table.
        </xs:documentation>
        </xs:annotation>
        </xs:pattern>
        <xs:pattern value="![0-9]+">
        <xs:annotation>
        <xs:documentation>
            This paging mode will display at most a fixed number of values. If the
table have more lines, the
            user
            will have to refine its search first.
        </xs:documentation>
        </xs:annotation>
        </xs:pattern>
    </xs:restriction>
</xs:simpleType>

<!-- ***** -->
<!-- * FORMULAS * -->
<!-- ***** -->

<!-- $Id: XmlTree1-Formula.xsd 43796 2013-09-20 21:19:28Z jbsilvy $ -->

<xs:complexType name="Formula">
    <xs:sequence maxOccurs="1" minOccurs="1">
        <xs:element maxOccurs="unbounded" name="setting" type="xt:FormulaSettingDefinition"
minOccurs="0"></xs:element>
        <xs:element maxOccurs="unbounded" name="parameter" type="
xt:FormulaParameterDefinition" minOccurs="0"></xs:element>
        <xs:element maxOccurs="unbounded" name="result" type="xt:FormulaResultDefinition"
minOccurs="0"></xs:element>
    </xs:sequence>
    <xs:attribute name="formulaId" use="required" type="xs:string"></xs:attribute>
</xs:complexType>

<xs:complexType name="FormulaSettingDefinition">

```

```

    <xs:attribute name="name" use="required" type="xs:string"></xs:attribute>
    <xs:attribute name="value" use="required" type="xs:string"></xs:attribute>
  </xs:complexType>

  <xs:complexType abstract="true" name="FormulaParameterDefinition">
    <xs:annotation>
      <xs:documentation>
        The name of the result. If it is omitted, the default result name will be used.
      </xs:documentation>
    </xs:annotation>
    <xs:attribute name="name" use="optional" type="xs:string"></xs:attribute>
    <xs:attribute name="properties" use="optional" type="xs:string"></xs:attribute>
  </xs:complexType>

  <xs:complexType name="ConstantFormulaParameterDefinition">
    <xs:complexContent>
      <xs:extension base="xt:FormulaParameterDefinition">
        <xs:attribute name="value" use="required" type="xs:float"></xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="EmptyFormulaParameterDefinition">
    <xs:complexContent>
      <xs:extension base="xt:FormulaParameterDefinition"></xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="FilterFormulaParameterDefinition">
    <xs:complexContent>
      <xs:extension base="xt:FormulaParameterDefinition">
        <xs:attribute name="filter" use="required" type="xs:string"></xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="PropertyFormulaParameterDefinition">
    <xs:complexContent>
      <xs:extension base="xt:FormulaParameterDefinition">
        <xs:attribute name="property" use="required" type="xs:string"></xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

```

```

    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="ResultFormulaParameterDefinition">
    <xs:complexContent>
      <xs:extension base="xt:FormulaParameterDefinition">
        <xs:attribute name="result" use="required" type="xs:string"></xs:attribute>
        <xs:attribute default="children" name="from" use="optional" type="
xt:resultSource"></xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="CombinedFormulaParameterDefinition">
    <xs:complexContent>
      <xs:extension base="xt:FormulaParameterDefinition">
        <xs:sequence maxOccurs="unbounded" minOccurs="0">
          <xs:element name="parameter" type="xt:FormulaParameterDefinition"></
xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="FormulaResultDefinition">
    <xs:annotation>
      <xs:documentation>
        A formula result describes a value returned by a formula.
      </xs:documentation>
    </xs:annotation>
    <xs:attribute name="name" use="optional" type="xs:string">
      <xs:annotation>
        <xs:documentation>
          This is the name of the result. The name of a result is what uniquely
identifies it. Two results
          with
          the same name will be considered the same.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute default="false" name="default" use="optional" type="xs:boolean">
      <xs:annotation>

```

```

        <xs:documentation>
            The default result is the value which should be used in aggregated report
            modes (pies, stacked
            graphs, ...). When no default result is selected, the default result of a
            node is the aggregation of all its
            selected values.
        </xs:documentation>
    </xs:annotation>
</xs:attribute>
<xs:attribute default="false" name="graphable" use="optional" type="xs:boolean">
    <xs:annotation>
        <xs:documentation>
            Determines wether this result should be displayed in graphs or not.
        </xs:documentation>
    </xs:annotation>
</xs:attribute>
</xs:complexType>

<xs:simpleType name="resultSource">
    <xs:annotation>
        <xs:documentation>
            For ResultFormulaParameterDefinition, it defines where we should look for the
            result.
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
        <xs:enumeration value="node">
            <xs:annotation>
                <xs:documentation>
                    The result comes from another formula on the same node.
                </xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="children">
            <xs:annotation>
                <xs:documentation>
                    The result comes from a formula on the child nodes.
                </xs:documentation>
            </xs:annotation>
        </xs:enumeration>
    </xs:restriction>
</xs:simpleType>

```

```
<xs:complexType name="HeatmapReport">
  <xs:sequence>
    <xs:element name="info" minOccurs="0">
      <xs:complexType>
        <xs:attribute name="properties" type="xs:string"/></xs:attribute>
      </xs:complexType>
    </xs:element>
    <xs:element name="x" type="xt:axisConf" minOccurs="0"/></xs:element>
    <xs:element name="y" type="xt:axisConf" minOccurs="0"/></xs:element>
  </xs:sequence>
  <xs:attribute name="order" type="xs:string"/></xs:attribute>
  <xs:attribute name="mode" type="xs:string"/></xs:attribute>
  <xs:attribute name="grid" type="xs:boolean"/></xs:attribute>
</xs:complexType>

<xs:complexType name="axisConf">
  <xs:attribute name="durationExpression" use="optional" type="xs:string"/></xs:attribute>
  <xs:attribute name="samplingPeriodExpression" type="xs:string"/></xs:attribute>
</xs:complexType>

<xs:complexType name="Threshold">
  <xs:attribute name="name" use="required" type="xs:string"/></xs:attribute>
  <xs:attribute name="value" use="required" type="xs:double"/></xs:attribute>
  <xs:attribute name="scaled" use="optional" type="xs:boolean"/></xs:attribute>
  <xs:attribute name="color" use="required" type="xt:HexColor"/></xs:attribute>
</xs:complexType>

<xs:complexType name="Axis">
  <xs:attribute name="minValue" use="optional" type="xs:float"/></xs:attribute>
  <xs:attribute name="maxValue" use="optional" type="xs:float"/></xs:attribute>
  <xs:attribute name="scaleFactor" use="optional" type="xs:string"/></xs:attribute>
  <xs:attribute name="filterExpression" use="required" type="xs:string"/></xs:attribute>
</xs:complexType>
```

```

<xs:complexType name="MetricCustomization">
  <xs:attribute name="filter" use="required" type="xs:string"/></xs:attribute>
  <xs:attribute name="color" use="optional" type="xt:HexColor"/></xs:attribute>
  <xs:attribute name="dash" use="optional">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="\d+(\_\d+)*"/></xs:pattern>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="width" use="optional" type="xs:float"/></xs:attribute>
  <xs:attribute name="marker" use="optional">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="none"/></xs:enumeration>
        <xs:enumeration value="line"/></xs:enumeration>
        <xs:enumeration value="circle"/></xs:enumeration>
        <xs:enumeration value="diamond"/></xs:enumeration>
        <xs:enumeration value="square"/></xs:enumeration>
        <xs:enumeration value="triangle"/></xs:enumeration>
        <xs:enumeration value="triangle_down"/></xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="curve" use="optional">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="none"/></xs:enumeration>
        <xs:enumeration value="spline"/></xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>

<xs:complexType name="TopNReport">
  <xs:sequence>
    <xs:element name="section-properties" minOccurs="0">
      <xs:complexType>
        <xs:attribute name="properties" type="xs:string"/></xs:attribute>
      </xs:complexType>
    </xs:element>
  </xs:sequence>

```

```

    <xs:attribute name="sectionGrouping" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="stacked"></xs:enumeration>
          <xs:enumeration value="summary"></xs:enumeration>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="sectionOrdering" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="alphabetical"></xs:enumeration>
          <xs:enumeration value="decreasing"></xs:enumeration>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="maxNbSections" use="optional" type="xs:positiveInteger"></
xs:attribute>
    <xs:attribute name="othersColor" use="optional" type="xt:HexColor"></xs:attribute>
    <xs:attribute name="expandOthers" use="optional" type="xs:boolean"></xs:attribute>
  </xs:complexType>

  <xs:simpleType name="HexColor">
    <xs:restriction base="xs:string">
      <xs:pattern value="#[0-9a-f]{6}"></xs:pattern>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="StatusReport">
    <xs:sequence>
      <xs:element name="value" minOccurs="0">
        <xs:complexType>
          <xs:attribute name="show" type="xs:boolean"></xs:attribute>
          <xs:attribute name="rounding" type="xs:int"></xs:attribute>
          <xs:attribute name="scale" type="xs:string">
            <xs:annotation><xs:documentation>Deprecated</xs:documentation></
xs:annotation>
          </xs:attribute>

```

```

        <xs:attribute name="scaleFactor" type="xs:string">
          <xs:annotation><xs:documentation>Deprecated</xs:documentation></
xs:annotation>
        </xs:attribute>
      </xs:complexType>
    </xs:element>
    <xs:element name="marker" minOccurs="0">
      <xs:complexType>
        <xs:attribute name="type" type="xs:string"></xs:attribute>
        <xs:attribute name="size" type="xs:string"></xs:attribute>
      </xs:complexType>
    </xs:element>
    <xs:element name="info" minOccurs="0">
      <xs:complexType>
        <xs:attribute name="properties" type="xs:string"></xs:attribute>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="order" type="xs:string"></xs:attribute>
</xs:complexType>

<xs:complexType name="TreemapReport">
  <xs:attribute name="depth">
    <xs:simpleType>
      <xs:restriction base="xs:int">
        <xs:minInclusive value="1"></xs:minInclusive>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="size" type="xs:string"></xs:attribute>
  <xs:attribute name="color" type="xs:string"></xs:attribute>
</xs:complexType>

<!-- ***** -->
<!-- * WIRING * -->
<!-- ***** -->

```

```
<xs:complexType name="Wiring">
  <xs:sequence maxOccurs="1" minOccurs="1">
    <xs:element maxOccurs="unbounded" name="trigger" type="xt:WiringTrigger" minOccurs="0"/></xs:element>
    <xs:element maxOccurs="unbounded" name="adapter" type="xt:WiringAdapter" minOccurs="0"/></xs:element>
  </xs:sequence>
  <xs:attribute default="false" name="activated" use="optional" type="xs:boolean"/></xs:attribute>
</xs:complexType>

<!-- Trigger -->

<xs:complexType name="WiringTrigger">
  <xs:sequence maxOccurs="1" minOccurs="1">
    <xs:element maxOccurs="1" name="src-filter" type="xt:WiringFilter" minOccurs="1"/></xs:element>
    <xs:element maxOccurs="1" name="dest-filter" type="xt:WiringFilter" minOccurs="0"/></xs:element>
    <xs:element maxOccurs="1" name="action" type="xt:WiringAction" minOccurs="1"/></xs:element>
  </xs:sequence>
  <xs:attribute name="desc" use="required" type="xs:string"/></xs:attribute>
</xs:complexType>

<!-- Filters -->

<xs:complexType abstract="true" name="WiringFilter">
</xs:complexType>

<xs:complexType name="JsonQueryFilter">
  <xs:complexContent>
    <xs:extension base="xt:WiringFilter">
      <xs:attribute name="query" use="required" type="xs:string"/></xs:attribute>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Actions -->
```

```

<xs:complexType abstract="true" name="WiringAction">
</xs:complexType>

<xs:complexType name="PassthroughAction">
  <xs:complexContent>
    <xs:extension base="xt:WiringAction">
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="DrilldownAction">
  <xs:complexContent>
    <xs:extension base="xt:WiringAction">
      <xs:sequence maxOccurs="1" minOccurs="1">
        <xs:element maxOccurs="1" name="report" type="xt:WiringValueSelector"
minOccurs="1"></xs:element>
        <xs:element maxOccurs="1" name="query" type="xt:WiringValueSelector"
minOccurs="1"></xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="CustomAction">
  <xs:complexContent>
    <xs:extension base="xt:WiringAction">
      <xs:sequence maxOccurs="1" minOccurs="1">
        <xs:element maxOccurs="unbounded" name="transformation" type="
xt:WiringTransformation" minOccurs="1"></xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Adapter -->

<xs:complexType name="WiringAdapter">
  <xs:sequence maxOccurs="1" minOccurs="1">
    <xs:element maxOccurs="1" name="src-filter" type="xt:WiringFilter" minOccurs="1"></
xs:element>

```

```

        <xs:element maxOccurs="unbounded" name="transformation" type="
xt:WiringTransformation" minOccurs="1"></xs:element>
    </xs:sequence>
    <xs:attribute name="desc" use="required" type="xs:string"></xs:attribute>
</xs:complexType>

<!-- Transformations -->

<xs:complexType abstract="true" name="WiringTransformation">
</xs:complexType>

<xs:complexType name="AddTransformation">
    <xs:complexContent>
        <xs:extension base="xt:WiringTransformation">
            <xs:sequence maxOccurs="1" minOccurs="1">
                <xs:element maxOccurs="1" name="value" type="xt:WiringValueSelector"
minOccurs="1"></xs:element>
            </xs:sequence>
            <xs:attribute name="context" use="required" type="xs:string"></xs:attribute>
            <xs:attribute name="key" use="required" type="xs:string"></xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="RenameTransformation">
    <xs:complexContent>
        <xs:extension base="xt:WiringTransformation">
            <xs:attribute name="fromContext" use="required" type="xs:string"></xs:attribute>
            <xs:attribute name="fromKey" use="required" type="xs:string"></xs:attribute>
            <xs:attribute name="toContext" use="required" type="xs:string"></xs:attribute>
            <xs:attribute name="toKey" use="required" type="xs:string"></xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="CopyTransformation">
    <xs:complexContent>
        <xs:extension base="xt:WiringTransformation">
            <xs:attribute name="fromContext" use="required" type="xs:string"></xs:attribute>
            <xs:attribute name="fromKey" use="required" type="xs:string"></xs:attribute>
            <xs:attribute name="toContext" use="required" type="xs:string"></xs:attribute>
            <xs:attribute name="toKey" use="required" type="xs:string"></xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

```

```
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>

    <!-- Value selectors -->

    <xs:complexType abstract="true" name="WiringValueSelector">
    </xs:complexType>

    <xs:complexType name="StaticValueSelector">
      <xs:complexContent>
        <xs:extension base="xt:WiringValueSelector">
          <xs:attribute name="value" use="required" type="xs:string"/></xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="JsonQueryValueSelector">
      <xs:complexContent>
        <xs:extension base="xt:WiringValueSelector">
          <xs:attribute name="query" use="required" type="xs:string"/></xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="PropertyValueSelector">
      <xs:complexContent>
        <xs:extension base="xt:WiringValueSelector">
          <xs:attribute name="properties" use="required" type="xs:string"/></xs:attribute>
          <xs:attribute name="idPath" use="required" type="xs:string"/></xs:attribute>
          <xs:attribute name="queryPath" use="required" type="xs:string"/></xs:attribute>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>

  </xs:schema>
```

2.4 XmlReport1.xsd

```

<?xml version="1.0" encoding="UTF-8"?><schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:xr
="http://www.watch4net.com/APG/Web/XmlReport1" xmlns:xt="http://www.watch4net.com/APG/Web/
XmlTree1" xmlns:xmime="http://www.w3.org/2005/05/xmime" xmlns:xsd="http://www.w3.org/2001/
/XMLSchema" xmlns:Q1="http://www.watch4net.com/APG/Web/XmlReport1" targetNamespace="http://www.
watch4net.com/APG/Web/XmlReport1" attributeFormDefault="unqualified" elementFormDefault="
qualified">

    <import schemaLocation="XmlTree1.xsd" namespace="http://www.watch4net.com/APG/Web/XmlTree1"
></import>

    <import schemaLocation="xmime.xsd" namespace="http://www.w3.org/2005/05/xmime"></import
>

    <element name="compound-element" type="xr:CompoundElement"></element>
    <element name="graph-element" type="xr:GraphElement"></element>
    <element name="table-element" type="xr:TableElement"></element>
    <element name="error-element" type="xr:ErrorElement"></element>
    <element name="image-element" type="xr:ImageElement"></element>

    <group name="ReportElementChoice">
        <choice>
            <element ref="xr:compound-element"></element>
            <element ref="xr:graph-element"></element>
            <element ref="xr:error-element"></element>
            <element ref="xr:image-element"></element>
            <element ref="xr:table-element"></element>
            <group ref="xr:ExternalReportElements"></group>
        </choice>
    </group>

    <group name="ExternalReportElements">
        <choice><element ref="xr:status-element"></element></choice>
    </group>

    <complexType abstract="true" name="ReportElement">
        <sequence>
            <element xmime:expectedContentTypes="image/*" maxOccurs="1" name="icon" type="
base64Binary" minOccurs="0"></element>
            <element maxOccurs="1" name="report-preferences" type="xt:ReportPreferences"
minOccurs="0"></element>

```

```

        <element maxOccurs="1" name="runtime-preferences" type="xt:RuntimePreferences"
minOccurs="0"></element>

        <element maxOccurs="1" name="description" type="string" minOccurs="0"></element>

        <element maxOccurs="1" name="properties" minOccurs="0">
            <complexType>
                <sequence>
                    <element maxOccurs="unbounded" name="property" minOccurs="1">
                        <complexType>
                            <simpleContent>
                                <extension base="string">
                                    <attribute name="name" type="string"></attribute>
                                </extension>
                            </simpleContent>
                        </complexType>
                    </element>
                </sequence>
            </complexType>
        </element>
    </sequence>
    <attribute name="name" use="required" type="string"></attribute>
    <attribute name="id" use="required" type="string"></attribute>
    <attribute name="type" use="required" type="string"></attribute>
</complexType>

<complexType name="GraphElement">
    <complexContent>
        <extension base="xr:ReportElement">
            <sequence>
                <element xmime:expectedContentTypes="image/*" maxOccurs="1" name="graph" type
="base64Binary" minOccurs="0"></element>

                <element maxOccurs="1" name="datasets" type="xr:GraphDatasets" minOccurs="0">
</element>

                <element maxOccurs="1" name="legend" type="xr:GraphLegend" minOccurs="0"></
element>

            </sequence>
        </extension>
    </complexType>

```

```

        </complexContent>
      </complexType>

      <complexType name="GraphDatasets">
        <sequence>
          <element maxOccurs="unbounded" name="dataset" minOccurs="0">
            <complexType>
              <sequence>
                <element maxOccurs="1" name="dd" minOccurs="1">
                  <complexType>
                    <sequence>
                      <element maxOccurs="unbounded" name="d" type="string"
minOccurs="0"></element>

                      </sequence>
                    </complexType>
                  </element>
                  <element maxOccurs="unbounded" name="dv" minOccurs="0">
                    <complexType>
                      <sequence>
                        <element maxOccurs="unbounded" name="v" type="anySimpleType"
minOccurs="0"></element>

                      </sequence>
                    </complexType>
                  </element>
                </sequence>
              </complexType>
            </element>
          </sequence>
        </complexType>

      <complexType name="GraphLegend">
        <sequence>
          <element maxOccurs="unbounded" name="variable" minOccurs="0">
            <complexType>
              <complexContent>
                <extension base="string">
                  <attribute name="selected" type="boolean"/></attribute>

                  <attribute name="color">
                    <simpleType>

```

```

        <restriction base="string">
            <pattern value="#[0-9a-f]{6}"></pattern>

        </restriction>
    </simpleType>
</attribute>
<attribute name="variable" type="string"></attribute>

    </extension>
</simpleContent>
</complexType>
</element>
</sequence>
</complexType>

<complexType name="TableElement">
    <complexContent>
        <extension base="xr:ReportElement">
            <sequence>
                <element maxOccurs="1" name="table" type="xr:Table" minOccurs="1"></element>

            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="Table">
    <sequence>
        <element maxOccurs="1" name="header" minOccurs="1">
            <complexType>
                <sequence>
                    <element maxOccurs="unbounded" name="th" type="string" minOccurs="0"></
element>

                </sequence>
            </complexType>
        </element>
        <element maxOccurs="1" name="data" minOccurs="1">
            <complexType>
                <sequence>

```

```
<element maxOccurs="unbounded" name="tr" type="xr:TableRow" minOccurs="0"
></element>
    </sequence>
  </complexType>
</element>
</sequence>
</complexType>

<complexType name="TableRow">
  <choice maxOccurs="unbounded" minOccurs="0">
    <element name="tv" type="xr:TableValue"></element>
    <element name="ts" type="xr:TableString"></element>
    <element name="te" type="xr:TableException"></element>
  </choice>
  <attribute name="id" use="optional" type="string"></attribute>
</complexType>

<complexType name="TableValue">
  <simpleContent>
    <extension base="string">
      <attribute name="state" use="optional">
        <simpleType>
          <restriction base="string">
            <enumeration value="ok"></enumeration>
            <enumeration value="major"></enumeration>
            <enumeration value="critical"></enumeration>
          </restriction>
        </simpleType>
      </attribute>
      <attribute name="numeric-value" type="decimal"></attribute>
    </extension>
  </simpleContent>
</complexType>

<complexType name="TableString">
  <simpleContent>
    <extension base="string"></extension>
  </simpleContent>
</complexType>

<complexType name="TableException">
  <simpleContent>
```

```

        <extension base="string">
            <attribute name="reason" use="required">
                <simpleType>
                    <restriction base="string">
                        <enumeration value="restricted"></enumeration>
                        <enumeration value="error"></enumeration>
                        <enumeration value="missing"></enumeration>
                        <enumeration value="cantretrieve"></enumeration>
                    </restriction>
                </simpleType>
            </attribute>
        </extension>
    </simpleContent>
</complexType>

<complexType name="CompoundElement">
    <complexContent>
        <extension base="xr:ReportElement">
            <sequence maxOccurs="unbounded" minOccurs="0">
                <group ref="xr:ReportElementChoice"></group>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="ImageElement">
    <complexContent>
        <extension base="xr:ReportElement">
            <sequence>
                <element xmlns:xmime="http://www.w3.org/2003/11/xmime" expectedContentTypes="image/*" maxOccurs="1" name="image" type="base64Binary" minOccurs="1"></element>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="ErrorElement">
    <complexContent>
        <extension base="xr:ReportElement">
            <sequence>
                <element maxOccurs="1" name="message" type="string" minOccurs="0"></element>
            </sequence>
        </extension>
    </complexContent>
</complexType>

```

```

        <element name="stacktrace" type="string"></element>

    </sequence>
</extension>
</complexContent>
</complexType>

<element name="status-element" type="xr:StatusReportElement"></element>

<complexType name="StatusReportElement">
    <complexContent>
        <extension base="xr:ReportElement">
            <sequence>
                <element maxOccurs="1" name="status" type="xr:Status" minOccurs="0"></element>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="Status">
    <sequence>
        <element maxOccurs="unbounded" name="li" type="xr:StatusItem" minOccurs="1"></
element>
    </sequence>
    <attribute name="marker-type" type="string"></attribute>
    <attribute name="marker-size" type="string"></attribute>
</complexType>

<complexType name="StatusItem">
    <sequence>
        <element name="name" type="string"></element>
        <element name="value" type="xr:Value"></element>
        <element maxOccurs="1" name="properties" minOccurs="0">
            <complexType>
                <sequence>
                    <element maxOccurs="unbounded" name="property" minOccurs="1">

```

```

        <complexType>
          <simpleContent>
            <extension base="string">
              <attribute name="name" type="string"/></attribute>
            </extension>
          </simpleContent>
        </complexType>
      </element>
    </sequence>
  </complexType>
</element>
</sequence>
<attribute name="id" use="optional"/></attribute>
</complexType>

<complexType name="Value">
  <simpleContent>
    <extension base="string">
      <attribute name="threshold" type="string"/></attribute>
    </extension>
  </simpleContent>
</complexType>
</schema>

```

2.5 xmlmime.xsd

```

<?xml version="1.0"?>
<!--
W3C XML Schema defined in the Describing Media Content of Binary Data in XML
specification
  http://www.w3.org/TR/xml-media-types

Copyright © 2005 World Wide Web Consortium,

(Massachusetts Institute of Technology, European Research Consortium for
Informatics and Mathematics, Keio University). All Rights Reserved. This
work is distributed under the W3C® Software License [1] in the hope that
it will be useful, but WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

```

```
[1] http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231

$Id: xmlmime.xsd,v 1.1 2005/04/25 17:08:35 hugo Exp $
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xmime="http://www.w3.org/2005/05/
xmlmime" targetNamespace="http://www.w3.org/2005/05/xmlmime">

  <xs:attribute name="contentType">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="3"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>

  <xs:attribute name="expectedContentTypes" type="xs:string"/>

  <xs:complexType name="base64Binary">
    <xs:simpleContent>
      <xs:extension base="xs:base64Binary">
        <xs:attribute ref="xmime:contentType"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="hexBinary">
    <xs:simpleContent>
      <xs:extension base="xs:hexBinary">
        <xs:attribute ref="xmime:contentType"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

</xs:schema>
```