



VAMDC-TAP service validation tool

Document Information

Editors: M.Doronin, L. Nenadovic
Authors: M.Doronin
Contributors:
Type of document: software documentation
Status: draft
Distribution: public
Work package: WP5
Version: 11.05
Date: 21/06/2011
Document code:
Document URL: http://www.vamdc.org/documents/software/TAPValidator_v11.05.pdf

Abstract: TAPValidator is a tool created to simplify and speed-up VAMDC-TAP node software development and deployment. It also enables the automated validation and monitoring of deployed VAMDC-TAP nodes. TAPValidator is a cross-platform application written in Java. This document covers the user interface and operation and also gives brief description of the application internal structure, modules and their interaction.

Version History

Version	Date	Modified By	Description of Change
V11.05	20/06/2011	M.Doronin	first draft

Disclaimer

The information in this document is subject to change without notice. Company or product names mentioned in this document may be trademarks or registered trademarks of their respective companies.

All rights reserved

The document is proprietary of the VAMDC consortium members. No copying or distributing, in any form or by any means, is allowed without the prior written agreement of the owner of the property rights.

This document reflects only the authors' view. The European Community is not liable for any use that may be made of the information contained herein.

Acknowledgements

VAMDC is funded under the "Combination of Collaborative Projects and Coordination and Support Actions" Funding Scheme of The Seventh Framework Program. Call topic: INFRA-2008-1.2.2 Scientific Data Infrastructure. Grant Agreement number: 239108.

CONTENTS

1	Graphical user interface	2
1.1	Menu and query	3
1.2	Restrictables keywords	3
1.3	VAMDC-XSAMS Document	3
1.4	Blocks locator	4
1.5	Status panel	4
1.6	Validation panel	4
2	Configuration	5
2.1	Local Settings	5
2.2	Network settings	6
2.3	Plugin mode settings	7
2.4	Control buttons	7
3	Command-Line mode	8
3.1	Command-line operation	9
3.2	Report file format	9
4	TAPValidator developer documentation	10
4.1	Structure diagram	10
4.2	Modules description	11
4.3	Modules interaction	12

TAPValidator is an application that simplifies the development and verification of TAP-VAMDC services.

XSAMS documents can be sourced from:

- remote TAP-VAMDC services;
- plugin for Java TAP-VAMDC implementation;
- local files.

Application supports both *Graphical user interface* and *Command-Line mode* operation. In command-line mode document validation results are saved in specific XML format, defined by the report.xsd schema, provided in sources.

GRAPHICAL USER INTERFACE

By default, TAPValidator starts in graphical mode.

When first run, it is necessary to modify the configuration settings. See *Configuration*.

Then, the usual operation would be to do several queries on the node (network/plugin) and, if an errors occurs, modify node software code. Iterate until documents are validating without errors.

- Main TAPValidator window

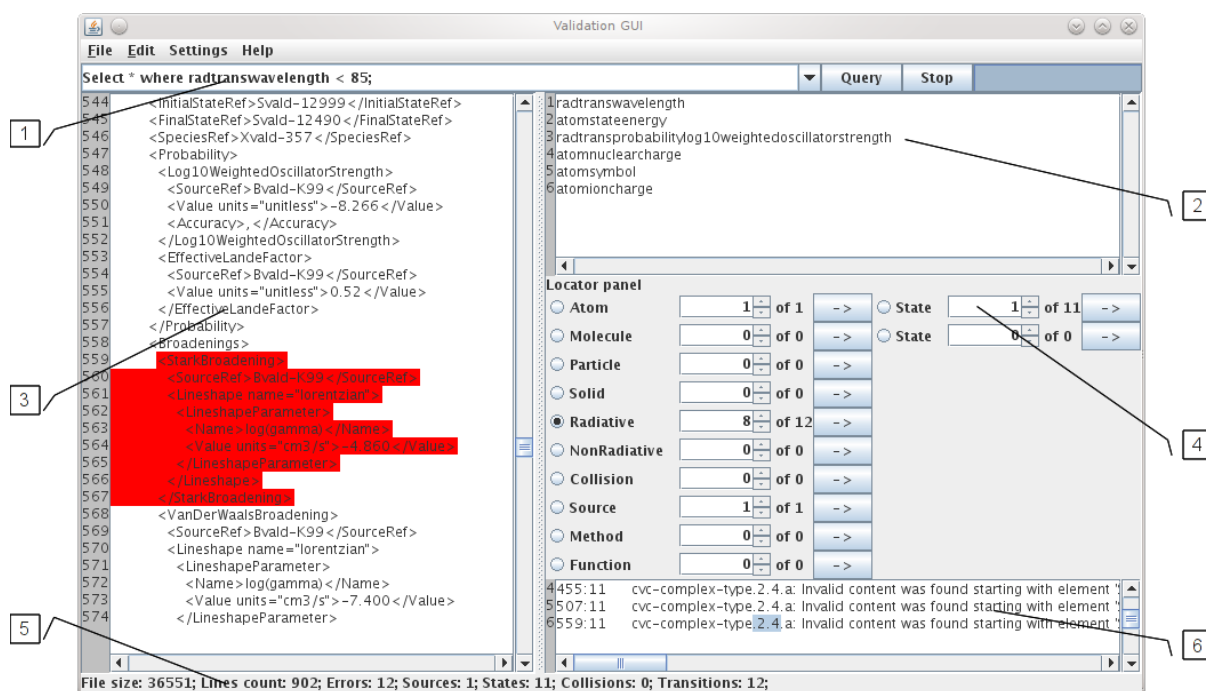


Figure 1.1: Main window

Most of the time you will be using the main window.

It has several important areas:

1. *Menu and query*
2. *Restrictables keywords*
3. *VAMDC-XSAMS Document*
4. *Blocks locator*
5. *Status panel*
6. *Validation panel*

1.1 Menu and query

This area gives the control over program.

1.1.1 Menu

Menu has the following structure

- File
 - Open
 - Save
 - Exit
- Edit
 - Find
 - Find Next
- Settings
 - Configure
- Help
 - About

Each menu item has its own shortcut keys combination.

You may load or save files while working in any operation mode.

1.1.2 Query

Query panel has a selector with last 10 successful queries, Query, Stop buttons and progress indicator.

Since document size is unknown during download, progress indicator just steps every 5000 lines of incoming XML document and gets full when document is fully loaded and processed.

If any error occurs during query, error message appears, then stack trace is printed to the stdout for the detailed information.

1.2 Restrictables keywords

This list displays a list of Restrictables keywords supported by the node.

Double-click on a line will add corresponding keyword to the end of query string.

1.3 VAMDC-XSAMS Document

This panel holds the VAMDC-XSAMS document, opened from file or returned by node.

Double-click on a line centers on it.

Located blocks and search results are highlighted by gray color, elements with validation errors are highlighted with red.

1.4 Blocks locator

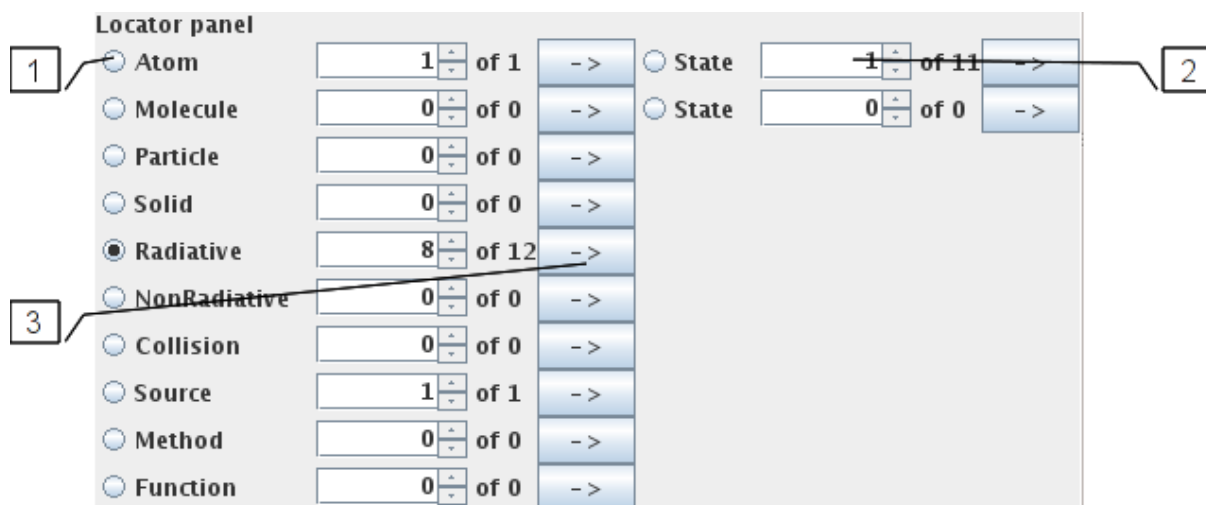


Figure 1.2: Block locator panel functions

Locator panel allows quick browsing through document sections.

1. Active section indicates that this was the last read/last seeked section.

Activate any inactive section to jump to current block index of that type.

2. Block index selector.

Allows to jump to a block with selected number in order.

3. Jump to next block button

Pressing that button would move you to the next block of that type starting from the current position in VAMDC-XSAMS document. If no blocks of this type are present latter in document, you will be directed to the first block of that type.

1.5 Status panel

Displays some document metrics, or in case of error occurred, error description.

1.6 Validation panel

For each of the validation errors displays position in document and error description.

Double-click on any line will scroll XSAMS document to selected error and highlight element that contains error.

CONFIGURATION

TAPValidator configuration window has three sections:

- *Local Settings*
- *Network settings*
- *Plugin mode settings*

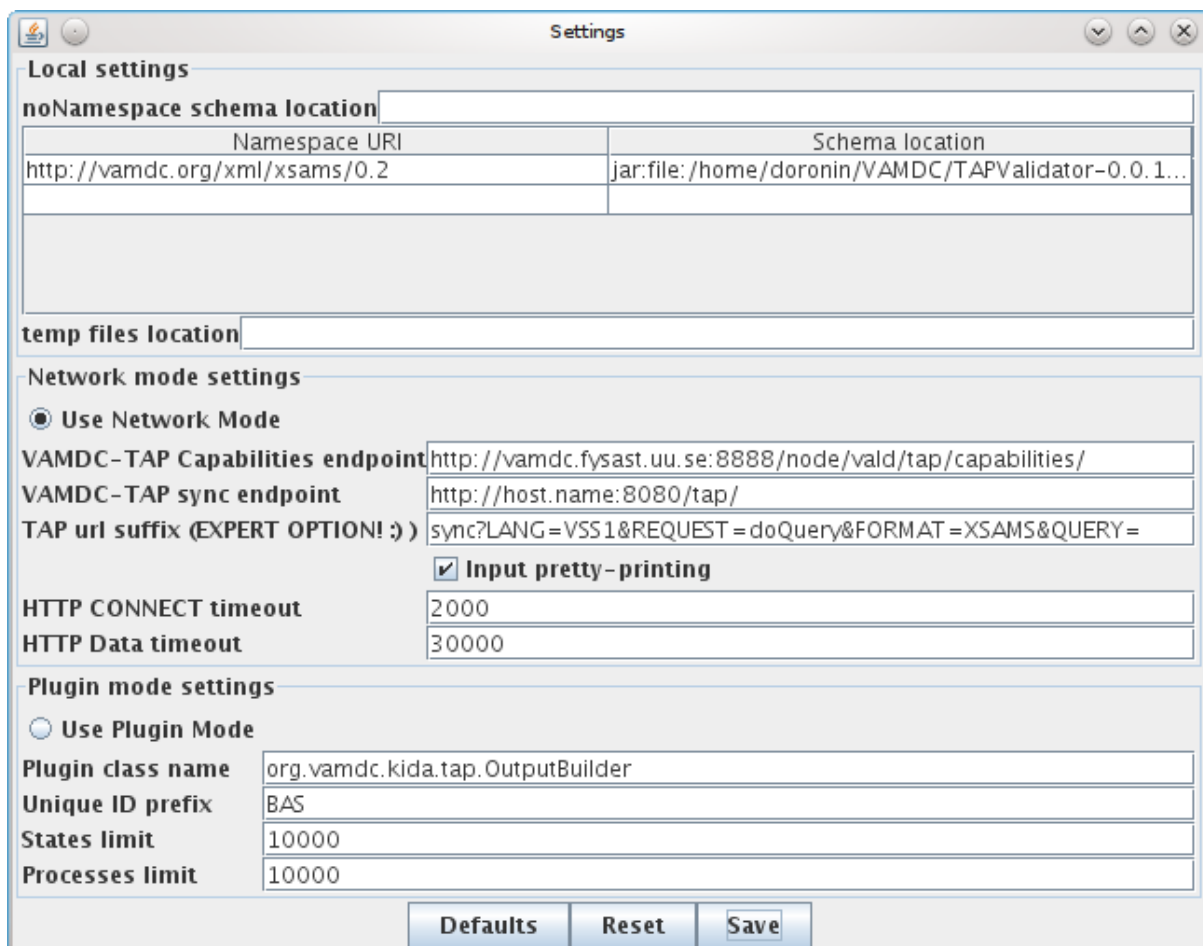


Figure 2.1: Settings window

2.1 Local Settings

In this section you may configure options, specific to your installation:

- **noNamespace schema location**

File containing schema for elements of instance document without namespace specified. In this field you must specify path to xsams.xsd on your computer, if you intend to validate documents against old versions of schema.

Double-click on the field to open file open dialog.

When filled, field is validated to check if specified file exists and is readable.
- **Table with pairs of namespace uri and schema locations** By default it is filled with location of bundled schema file.

To add new namespace, double-click on either of two cells in the end of the table, fill in the data.

To delete namespace, remove contents of either of corresponding cells, the row will be removed automatically.
- **Temporary files location** If system temporary directory is inaccessible for some reason, specify here a writeable directory where to put temporary files.

Double-click on the field to open directory choice dialog.

When filled, field is validated to check if specified directory exists and the user has write premissions.

2.2 Network settings

This mode may be used for testing of local deployment of TAP-VAMDC node software, or to verify operation of remote node.

- **VAMDC-TAP Capabilities endpoint** If you use Python/Django node software or Java implementation of one, all what you need to specify is TAPService capabilities endpoint URL. Validation tool will automatically retrieve TAP endpoint address from it. Also for each query it will be checking that availability endpoint tells that service is available.
- **VAMDC-TAP sync endpoint** If for some reason TAP endpoint url reported in your service capabilities is inaccessible, you may specify it in this field.

WARNING: To make program work with this endpoint, you need to erase everything from the “Capabilities endpoint” field.
- **TAP URL suffix EXPERT OPTION** In this field you may modify TAP URL suffix. That is a part, defined by TAP-VAMDC service specification. Default value should work, so do not change it unless specifically instructed by node software provider or revised standard specification.
- **Input pretty-printing** Mark this checkbox if you wish TAPValidator to pretty-print(reformat) input XML. If unchecked, document is treated and displayed as it is provided by TAP-VAMDC node.

WARNING: pretty-printing normally doesn't change any values or elements order, though it may change backslashed quotes and triangle braces in attribute values into " and < or > respectively. This is not a bug but rather a feature.
- **HTTP Connect timeout** HTTP connection establishing timeout, in milliseconds. Default value should be fine.
- **HTTP Data timeout** Receiving data wait timeout, in milliseconds. Increase this value if it takes too long for node software to respond and TAPValidator times out.

2.3 Plugin mode settings

This mode is used for testing and development of Java TAP-VAMDC node software plugins. To use this mode, your plugin JAR or classes must be in classpath and be accessible.

- **Plugin class name** Fully qualified name of the class implementing *org.vamdc.tapservice.api.DatabasePlug* interface.
- **Unique ID prefix** Prefix for identifiers produced by *XSAMSLibrary.BuildIDs* class
- **States and Processes limit** Maximum amount of states and processes included in produced document. After reaching that limit, *XSAMSLibrary.XSAMSDaType* class will refuse to add any states/processes to output document.

2.4 Control buttons

- **Defaults** Reset all configuration options to their default values.
- **Reset** Reload all fields with current effective configuration parameters.
- **Save** Save modified configuration. Will display an error if something went wrong while applying new configuration.

WARNING: For configuration to take effect, it is necessary to press the save button, closing the window will not apply the changes.

COMMAND-LINE MODE

To use TAPValidator in automated tests of TAP-VAMDC nodes, command-line mode can be employed.

To get full list of supported options, call program with “-h” option:

```
$ java -jar target/TAPValidator-0.0.1-SNAPSHOT-jar-with-dependencies.jar -h
```

You will get a list of options with their descriptions, like:

```
Usage: prog [options]
```

```
Available options:
```

```
-u, --tap_url "value" : TAP-VAMDC service TAP endpoint URL  
    e.g. -u http://service:port/TAP/  
    -u http://service:port/tap/
```

```
-c, --capabilities_url "value" : TAP-VAMDC service capabilities endpoint URL,  
    e.g. -c "http://service:port/VOSI/capabilities"  
    or -c "http://service:port/tap/capabilities/"  
    Only first specified service will be used.
```

```
-p, --prettyprint : Re-format input XML
```

```
-t, --temp_dir "value" : Directory for temporary files.  
    Usually, system default is used.  
    If this directory is not writeable, using memory storage
```

```
-s, --schemalocation "value" : No namespace Schema location to validate output against
```

```
-n, --nsschemalocation "value" : Space-separated list of pairs of namespace url and  
    relevant schema location to validate output against
```

```
-q, --query "value" : VSS1/VSS2 Query to send to service.  
    Can be specified multiple times to do multiple queries on the same tapservice
```

```
-o, --output_dir "value" : Folder where to save result files.  
    Needs to be specified to initiate command-line mode
```

```
-h, --help : Print usage and exit
```

All options except -h and -o can be specified both in GUI and in command-line mode, option -q in GUI mode has no effect.

If option is specified in GUI mode, it overrides the specific setting, so, for example, multiple instances of validator may be called simultaneously to test several services and compare results.

3.1 Command-line operation

To initiate command-line mode, you need to specify output directory for resulting documents and reports and at least one query.

For each query validator saves document it got from service, in filename xsams(N).xml and validation report for it in report(N).xml, where N is an incrementing index, starting from 0 on each run.

WARNING! Validator will deny to overwrite any files, so before running it, make sure that output directory doesn't contain files from previous run.

3.2 Report file format

TAPValidator generates reports in its own XML format, defined by report.xsd schema. Report documents look like:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<report xmlns="http://vamdc.org/tapservice/validator/report">
<nodeCapabilitiesUrl>http://vamdc.fysast.uu.se:8888/node/vald/tap/capabilities/
</nodeCapabilitiesUrl>
<nodeTapSyncUrl>http://host.name:8080/tap/</nodeTapSyncUrl>
<nodeAvailable>true</nodeAvailable>
<queryString>Select * where radtranswavelength &lt; 85;</queryString>
<queryDate>2011-04-08+02:00</queryDate>
<documentInfo rowCount="902" size="36551">
  <documentFileName>xsams0.xml</documentFileName>
  <blockCount type="atom">1</blockCount>
  <blockCount type="atomState">11</blockCount>
  <blockCount type="molecule">0</blockCount>
  <blockCount type="moleculeState">0</blockCount>
  <blockCount type="particle">0</blockCount>
  <blockCount type="solid">0</blockCount>
  <blockCount type="collision">0</blockCount>
  <blockCount type="radiative">12</blockCount>
  <blockCount type="nonradiative">0</blockCount>
  <blockCount type="source">1</blockCount>
  <blockCount type="method">0</blockCount>
  <blockCount type="function">0</blockCount>
</documentInfo>
<validationError endCol="29" endRow="307" startCol="11" startRow="299">
  <elementName>StarkBroadening</elementName>
  <errorText>cvc-complex-type.2.4.a: Invalid content was found starting with
  element 'StarkBroadening'. One of '{Name, Comments, SourceRef, Broadening}'
  is expected.</errorText>
</validationError>
<validationError endCol="29" endRow="359" startCol="11" startRow="351">
  <elementName>StarkBroadening</elementName>
  <errorText>cvc-complex-type.2.4.a: Invalid content was found starting with
  element 'StarkBroadening'. One of '{Name, Comments, SourceRef, Broadening}'
  is expected.</errorText>
</validationError>
</report>
```

TAPVALIDATOR DEVELOPER DOCUMENTATION

This section describes the TAPValidator structure diagram, interaction of modules, packages. For further insight into modules operation see the interfaces javadoc and program sources.

4.1 Structure diagram

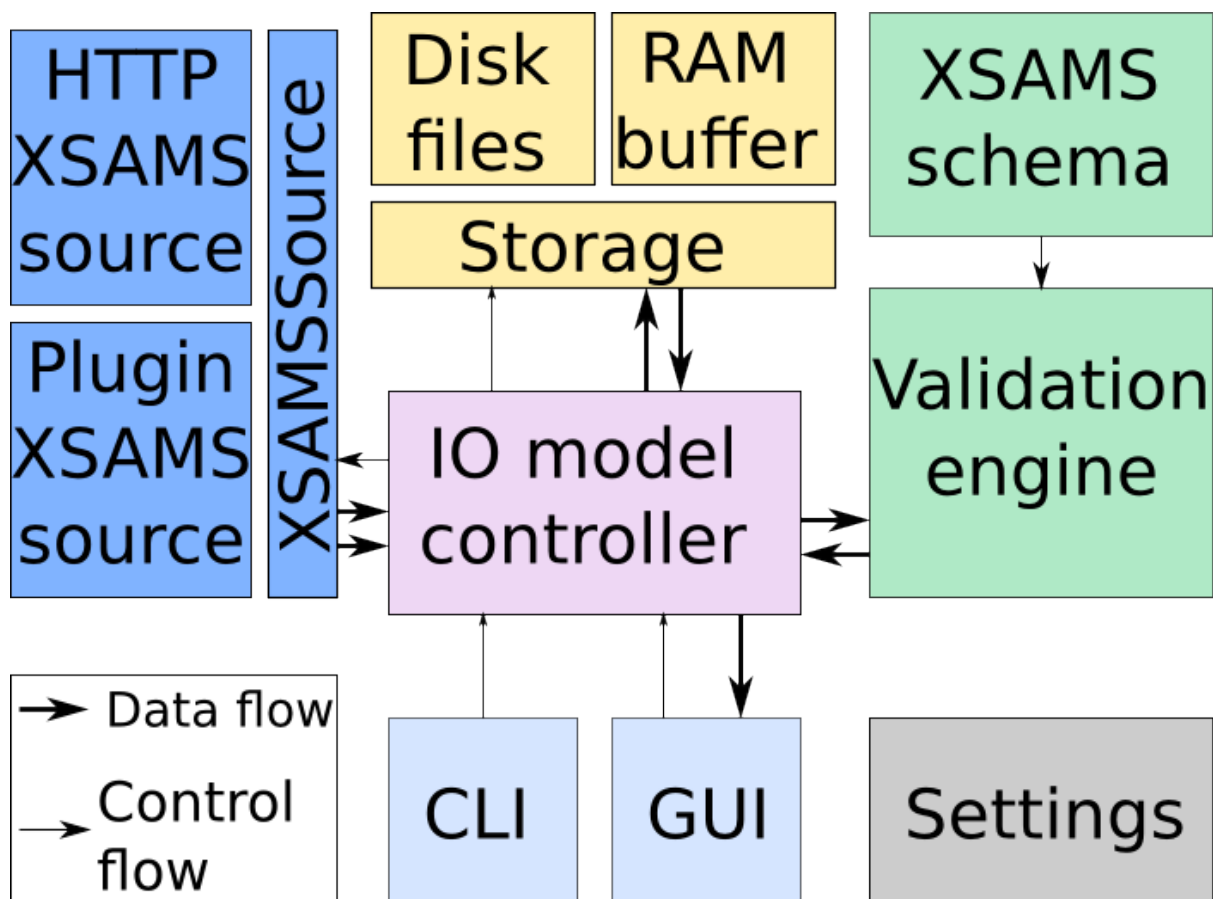


Figure 4.1: TAPValidator modules diagram

TAPValidator consists of the following functional blocks:

- IO controller, handling the operation,
- XSAMS XML stream sources,

- storage, capable of returning selected blocks of data,
- validation engine and blocks locator,
- CLI (command line interface) controller,
- GUI (graphical user interface) controller,
- settings storage

4.2 Modules description

4.2.1 IOController

IOController (class *org.vamdc.validator.iocontroller.XSAMSDocument*) manages all the application main functions, links all modules together and provides easy interfaces for CLI and GUI to control them.

4.2.2 XSAMSSource

XSAMSSource is able to provide the XML stream and an array of applicable Restrictable keywords. So far, two different implementations exists, **PluginXSAMSSource** and **HTTPXSAMSSource**, package *org.vamdc.validator.source*.

Plugin XSAMS source is created for the purpose of testing Java VAMDC-TAP implementation plugins without any infrastructure deployment. From the plugin point of view it looks identical to the VAMDC-TAP Java implementation, description of which is not a part of this document.

During the initialization **PliginXSAMSSource** instantiates a user-provided class, whose name is provided in settings, asks it for the supported restrictables and then waits for queries.

While processing a query, it repeats the functionality of the framework, then asks plugin to build XSAMS document, accepts JAXB structure for it and streams it into XML.

HTTP XSAMS source is able to query HTTP VAMDC-TAP services and is proven to work with all implementations that support sync TAP endpoint.

During the initialization it, through VOSI interfaces, checks the service availability, asks for supported restrictables, and then waits for queries.

During the query it checks again the availability, constructs the query URL and opens the stream containing the node response.

This source module also provides the functionality to pretty-print incoming XML stream and supports the decompression of a compressed stream.

4.2.3 Storage

Storage, with main class **RAWStorage** in package *org.vamdc.validator.storage*, is used to keep the copy of stream produced by XSAMSSource on disk, or, in case of a disk inaccessibility, in a memory buffer. It calculates document properties like size and line count, is able to copy document to arbitrary file and provides the random access method to read specific document lines or blocks of lines.

4.2.4 Validation engine

Validation engine, implemented in **org.vamdc.validator.validator.Validator.java**, uses Xerces2-j xml engine. During initialization it reads the schema, prepares the validation engine.

While validating the incoming XML stream it records each validation error and XML element in which it occurred, plus also it keeps track of positions of major XSAMS branches like species, states, process records and all the misc elements like Sources and Functions.

4.2.5 Settings

Located in the root package, *org.vamdc.validator.Settings* keeps all the configurable bits used by modules. HTTP VAMDC-TAP base URLs, plugin class, temp files location, schema location.

Parameter values are stored in Java Preferences subsystem.

Modification of the parameters is done either through the GUI settings dialog or through the CLI options.

4.3 Modules interaction

During the initialization IOController reads the settings, creates the desired XSAMS source, storage and initializes the validation engine.

During the query, IO Controller gives the query string to the source, passes the resulting stream to the validator, duplicating it to the storage, then when the query processing is done reports to the user interface.

During results display, IOController transparently gives out blocks of document from storage and collections of element and error locations from validator.

For file load operation, instead of asking the XSAMSSource module, IOController opens the file for stream input. The contents is processed the same way as in query procedure. It is also copied to the storage, so after the loading is finished there is no need for the source file to remain available.

For file save operation IOController asks storage to dump it's contents to a specific file.

4.3.1 CLI operation

Command-line interface is implemented in *org.vamdc.validator.cli.CLIProcess* class.

All required modules are instantiated once, queries are sent to the HTTPXSAMSSource, output is saved into destination folder, reports from all modules are combined into specific XML format and are saved in the same folder.

4.3.2 GUI operation

GUI is implemented in *org.vamdc.validator.gui.MainFrame* and its controller. Most of the main window panels are implemented in separate classes.

When initialized, GUI asks the IO controller for a list of restrictables, supported by selected source, displays it and waits for user operations.

During the query it periodically receives events from IO controller and updates the window, displaying actual status.

When settings are saved, GUI asks the IO controller to reinitialize all modules, and if any error occurs refuses to close the settings dialog, giving the error information.