CROSSXCHECK networks

User Guide

Version 7.0

Legal Marks

Crosscheck Networks
Needham, MA

Crosscheck Networks CloudPort™ Enterprise Version 4.0 User Guide, published September, 2019.

ASD-D-CP-138730

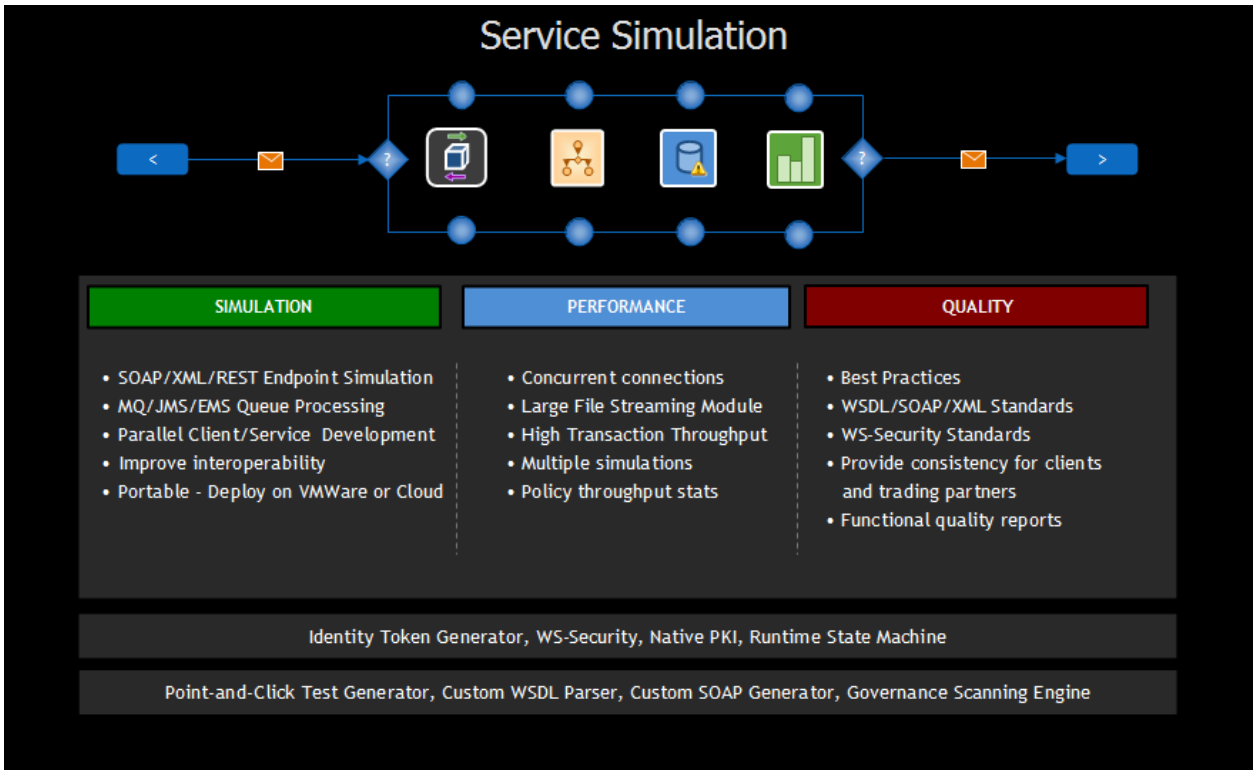
Table of Contents

INTRODUCTION

Welcome to the Crosscheck Networks CloudPort Service Simulation product. Crosscheck Networks created the CloudPort product to address the full range of web services simulation and client diagnostics features to compress the deployment lifecycle and allow parallel service and client development. With point-and-click service simulation, client development can be performed in parallel to service development and thus reduce the total project time.

Additional capabilities provide the means to validate the functional quality of the client requests, ensure that requests follow corporate best practices. CloudPort provides the portable framework that allows service simulations to be created and shared among team members to help ensure consistent development practices and techniques.

CloudPort can be used as a means to reduce sharing of IT and network resources for trading partner integration testing or new business unit integration. Simulations are portable which allows for integration testing locally before requiring shared lab time, firewall accessibility, and ultimately production server access.



LICENSING

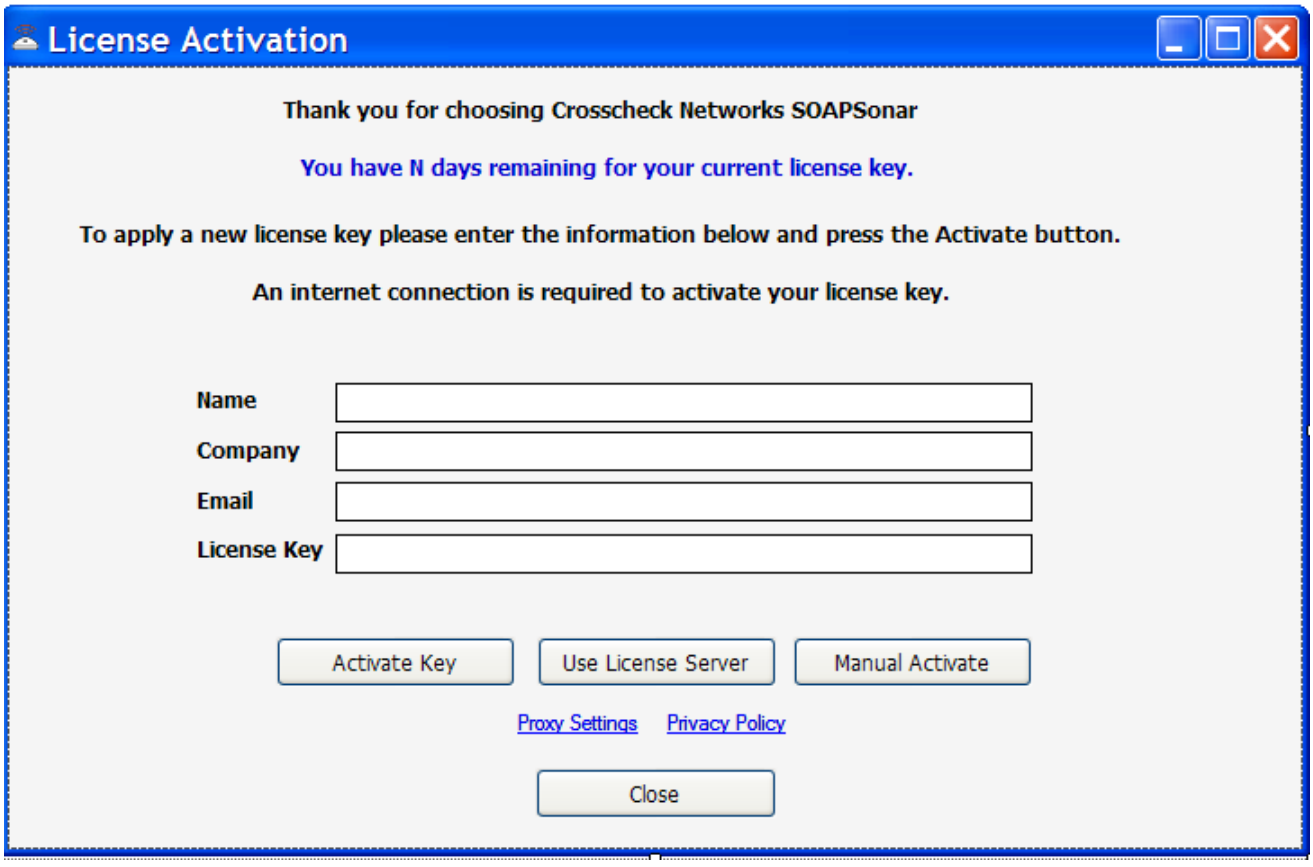
Licenses that are installed directly on the machine where CloudPort is running are called instance licenses. Instance licenses require activation when installed.

Licenses that are shared among any number of deployed CloudPort installations are called floating licenses. Floating licenses are leased for a period of time when needed, and returned to the pool of available licenses once the lease has expired.

INSTANCE BASED LICENSING

Instance based licensing uses license keys which are activated on the specific instance of the application. An instance license is enabled based on activation of the license key.

To apply an instance license, go to the **Registration->Install Instance License** menu and enter your user information as well as the license key provided to you.



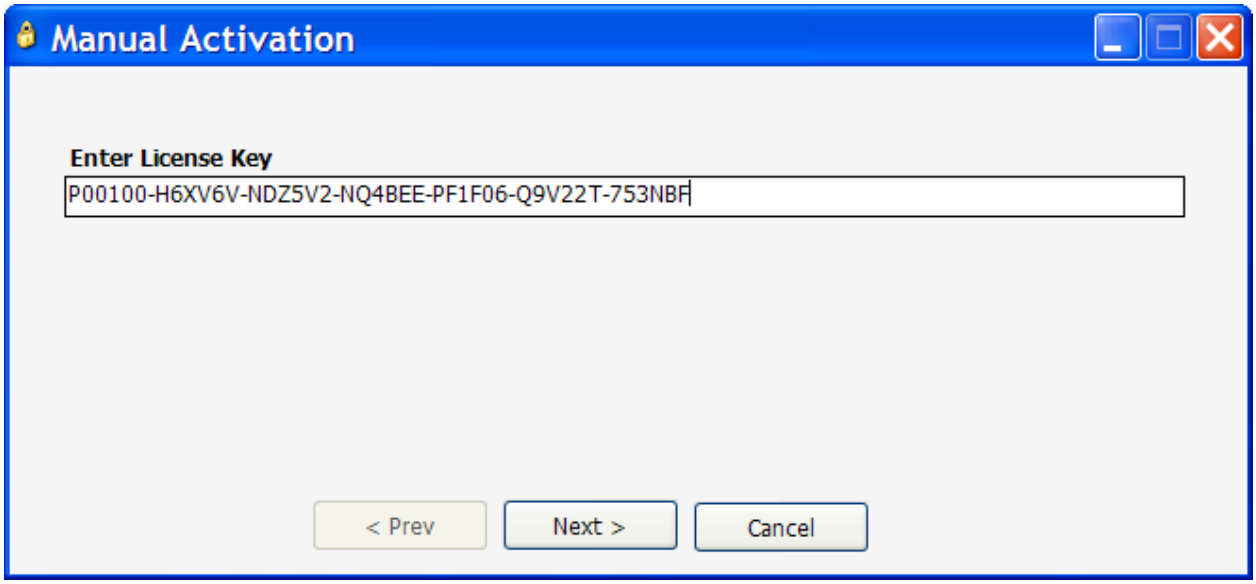
Internet License Activation

Once the information has been entered, press the Activate button to activate this license against the Crosscheck Networks Activation server. If the license key has valid activations remaining, your license will be enabled for this instance.
If you do not have internet access, refer to the Manual Activation steps shown in the Manual License Activation section below.

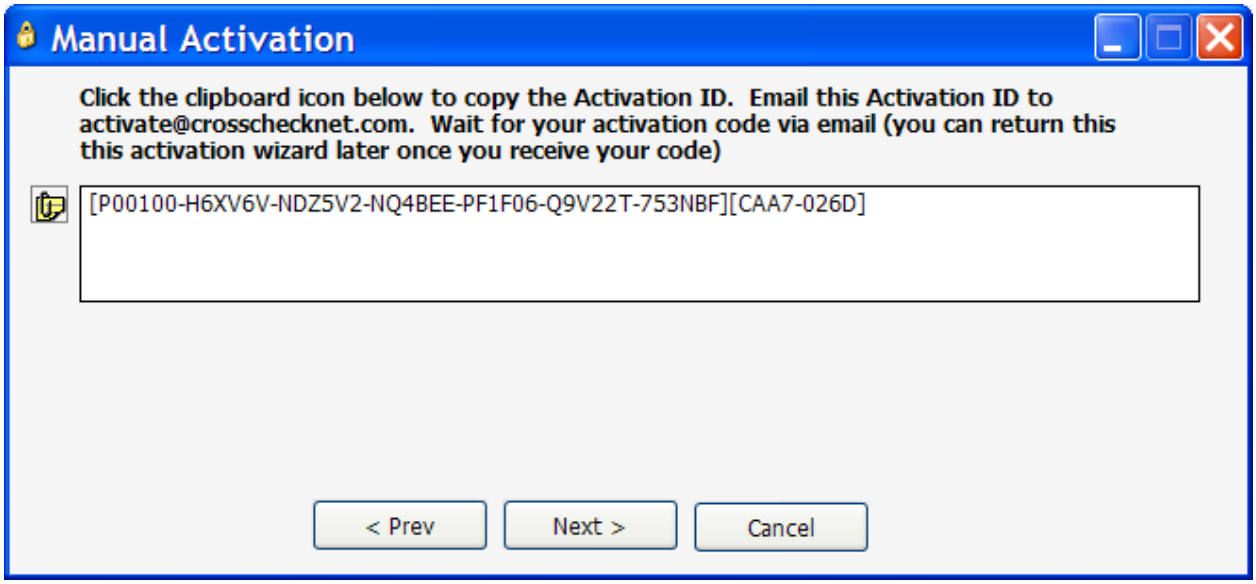
Manual License Activation

If you do not have internet access to activate your license using the Crosscheck Networks Activation server, then follow the steps below the manually activate your license instance.

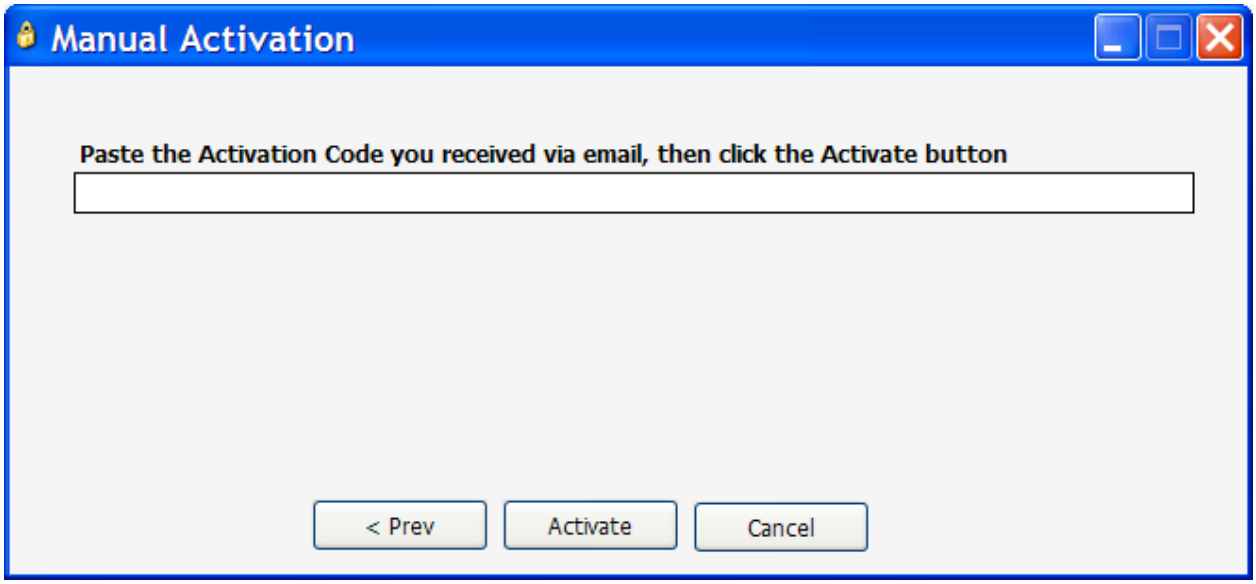
1. Enter you name, company name, email, and key code in the fields provided.
2. Click on the “Manual Activate” button below the key code field. The manual activation dialog will appear with your target license key to be activated. Please check this key to be sure it is the key you intend to activate. Press the Next button



3. The Manual activation code will be generated on the next screen. Copy the activation ID shown (click the clipboard button) and email this ID to activate@crosschecknet.com



4. Crosscheck Networks will return the Activation Code in an email. Enter the provided Activation Code in the “Enter Activation Code” box

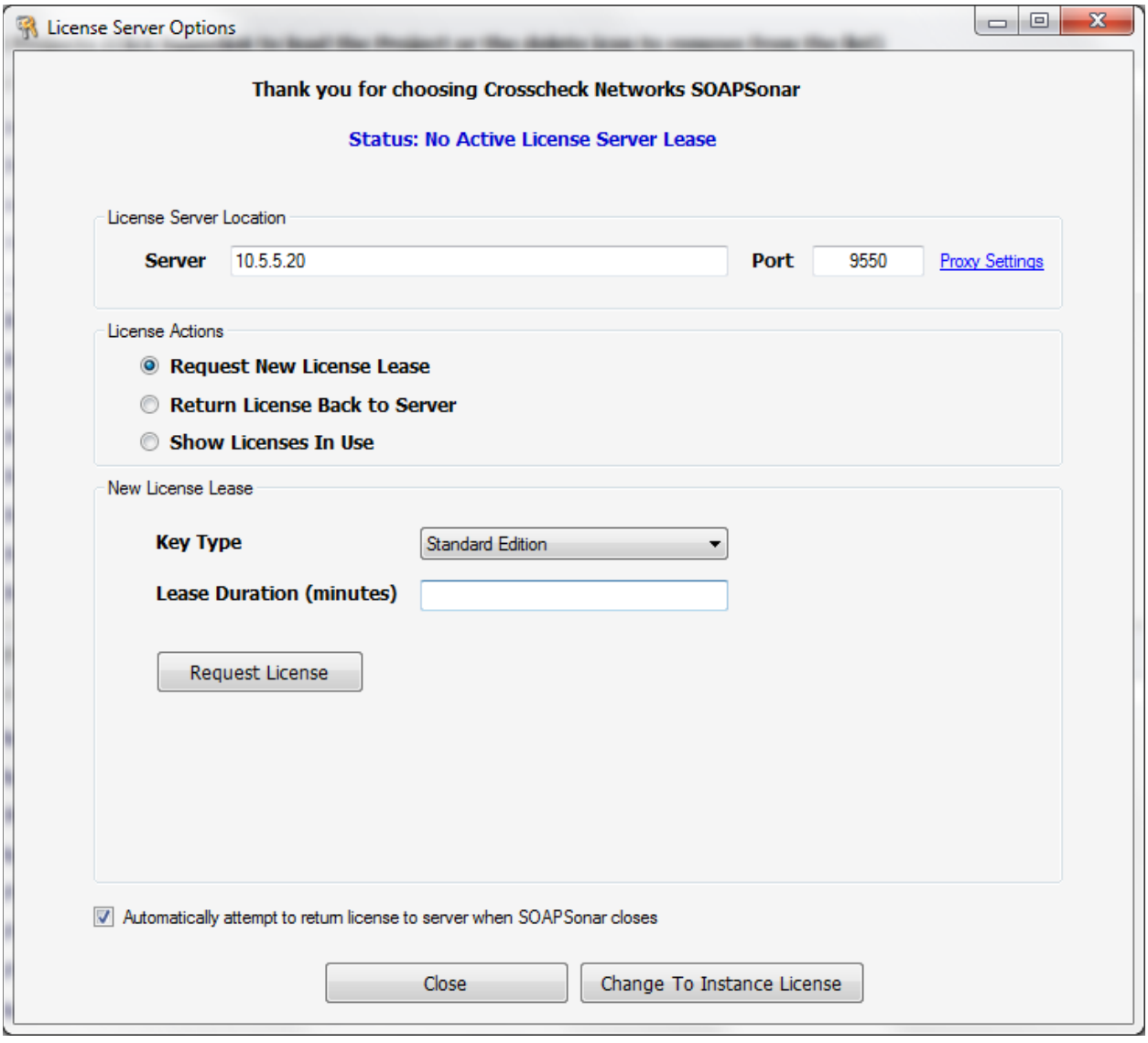


5. Close and restart the product.

FLOATING BASED LICENSING

Floating Licenses use a license server to obtain a lease for a license key for a specified duration. The Crosscheck Networks License Server is installed on some location within the network that can be accessed by the individual CloudPort instances. The Crosscheck Networks License Server uses a leasing model which does not require persistent network access to the licensing server, but rather only access at the time the lease is granted. Once a lease is granted, the product instance will not need to have a connection to the license server to use the leased license.

To request a new license lease, go to the **Registration->Use License Server** menu.

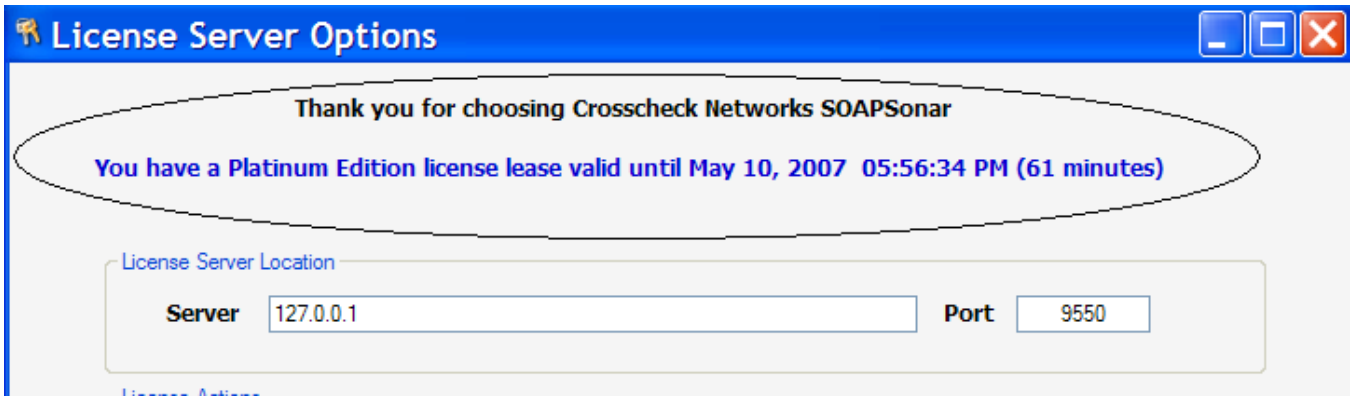


Request New License Lease

The floating license model works on a leasing system that only requires access to the license server at the time of the lease request. Licenses can thus be obtained and used for the duration of the lease interval without requiring access to the license server itself.

To request a new license lease, enter the information about the server location either as the domain name, or the IP address, and the port the server is configured to listen on (default is 9550). Then select the action "Request New License Lease" and choose the type of license and the duration of the lease and click on the Request License button to obtain a license lease from the server.

If successful, you will see the summary at the top of the screen that shows the duration of the newly obtained license lease and the type of license which is now active for use.



Return License Back to Server

You can choose to return a leased license back to the license server at any time by going to the "Return License Back to Server" option. If you do not have network access to the server you will be prompted with a release code that can be sent to the License Administrator for manually removing the lease from the server.

Show Licenses in Use

To see the current leases in effect for the selected license type and the duration of each, go to the "Show Licenses in Use" option and query the license server to see all active leases and the times remaining for each.

GETTING STARTED

This section will help explain the various aspects of the CloudPort work flow. Subtopics within this section include

- Understanding the CloudPort GUI
- Workflow Views
- Types of Simulations
- Capture a WSDL
- Auto-populate Settings for WSDL Simulations
- Working with Projects

Launch Options

Choose “Build Simulation” to launch the CloudPort Editor to build the rules for the simulation and save the information in a simulation project file.

Choose “Capture Live Traffic” to launch the Crosscheck Proxy Traffic Capture tool to capture live traffic requests and responses as well as timing. The captured transactions can be subsequently imported into CloudPort for automatically created simulations from the captured data.

Choose “Run Simulation” to launch the CloudPort Server to run the project rules and listen on the network for inbound client requests, or to a message queue for MQ or JMS simulations.

Note: If you are running the free runtime simulation player, there will be no launch option, it will automatically launch into the simulation project selection dialog to run the simulation.

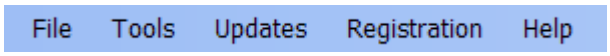
Understanding the CloudPort Editor

The CloudPort Editor interface is used to create the simulation rules used by the simulation server when running simulations. The interface is broken down into 3 views: Simulation Settings, Run View, and Report View. Menu options are available for accessing various tools and configuration options of the product. The CloudPort interface includes:

- Menu Items
- Simulation Settings
- Run View
- Report View

Menu Items

The CloudPort menu includes items allowing you to perform configuration setup, perform actions, and license the application.



File Menu: New WSDL Simulation
This option will open the WSDL capture panel and allow you to choose a network or file location where the WSDL resides to capture and parse into a CloudPort project for WSDL Service Simulation.

File Menu: New Custom Group
Creates a new Simulator group which can contain arbitrary XML simulation definitions.

File Menu: New SOAP Simulation
Creates a new SOAP Simulator for the selected WSDL operation which is used to identify inbound documents and the corresponding simulated responses.

File Menu: New XML Simulation
Creates a new XML Simulator which provides the means to identify inbound documents and provided any arbitrary simulated responses.

File Menu: Load Project
Loads a previously stored CloudPort project.

File Menu: Import – Proxy Server Traffic Capture
Runs the Crosscheck Proxy Server Traffic Capture tool enabling request and response transactions between client and servers to be captured and used for simulation creation.

File Menu: Save Project (As)
Stores current Project and Run settings for into a CloudPort project file.

File Menu: Clear Project
Removes all items in the Project

File Menu: Recent Projects
Load project by selecting from the provided list of recently loaded or saved projects.

File Menu: Recent WSDLs
Load WSDL by selecting from the provided list of recently captured WSDLs.

Simulation Menu: Deploy Simulation (Local)
Uses the current settings to run a live simulation using the simulation server.

Simulation Menu: Deploy Simulation (Local)
Enables deploying a simulation to a target server running the CloudPort Simulation Controller service

Tools Menu: PKI Management
Launch native PKI editor providing direct access to public-private keys for message signatures and encryption.

Tools Menu: UDDI Browser
Launches UDDI browser to search for services in a UDDI registry.

Tools Menu: XML Viewer
Provides means to view and edit XML data

Updates Menu: Check for Product Updates
Checks for updates to your existing installed CloudPort version

Registration Menu: Install Instance License
Used for licensing by the instance method where the installation of CloudPort is directly licensed on the machine where it is installed.

Registration Menu: Use License Server
Used to open the Floating License configuration panel to checkout a license from the license server, check-in a license to the license server, or query the license server for the current floating licenses in use by CloudPort instances on the network.

Registration Menu: Deactivation
For instance based licensing, this option deactivates a license and provides a deactivation ID which can be provided to activation@crosschecknet.com for moving the license to another machine.

Help Menu: Contents
Opens the online help.

Help Menu: Feedback
Send email to Crosscheck Networks regarding product feature request, bug reports, or general comments.

Help Menu: About
Shows license details and product version.

Project Editing

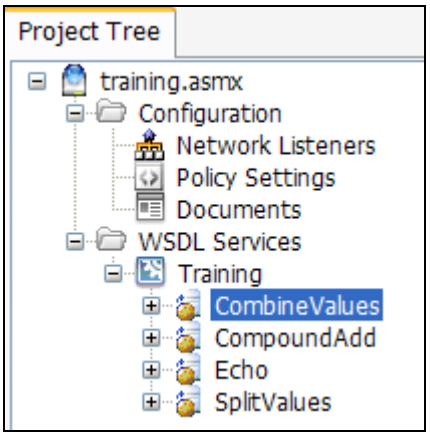
Shows the options and settings for creating Network endpoint and ESB-based simulations. This is also where document analysis such as WSDL Scoring, and building task rules or success criteria rules is performed.

Simulation Settings: Capture WSDL or OpenAPI

Used to obtain a WSDL or OpenAPI document from file or network location to parse and load the definitons for SOAP or OpenAPI testing

Simulation Settings: Project Tree

Shows the loaded simulation projects where inbound and outbound document rules are defined as well as functional success criteria characteristics of the inbound messages for each simulator.


Project Tree
training.asmx
Configuration
Network Listeners
Policy Settings
Documents
WSDL Services
Training
CombineValues
CompoundAdd
Echo
SplitValues

Simulation Settings – Simulator View

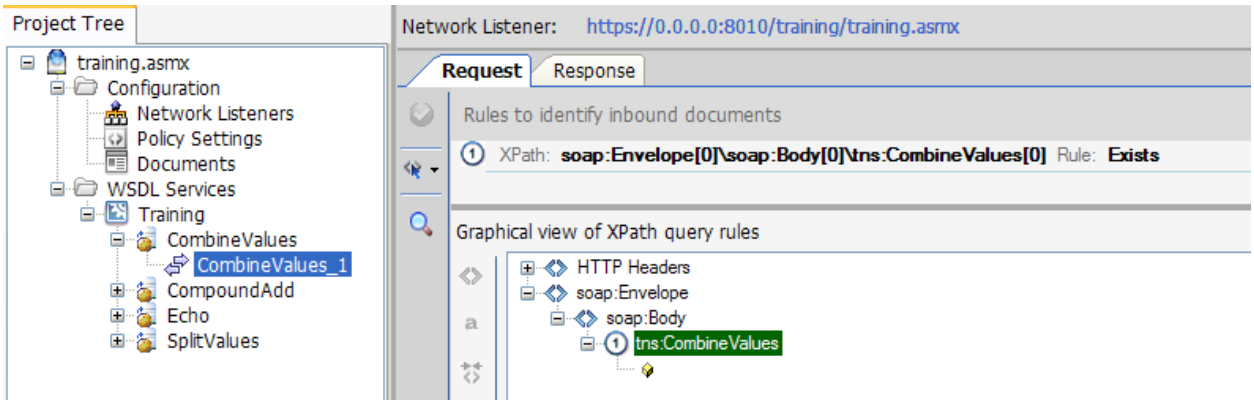
Displays request panel for defining simulator settings including:

Request Processing

Define inbound document identification, processing, and success rules. Sample documents can be imported from file, edited directly, or built using the SOAP generator based on XSD schema.

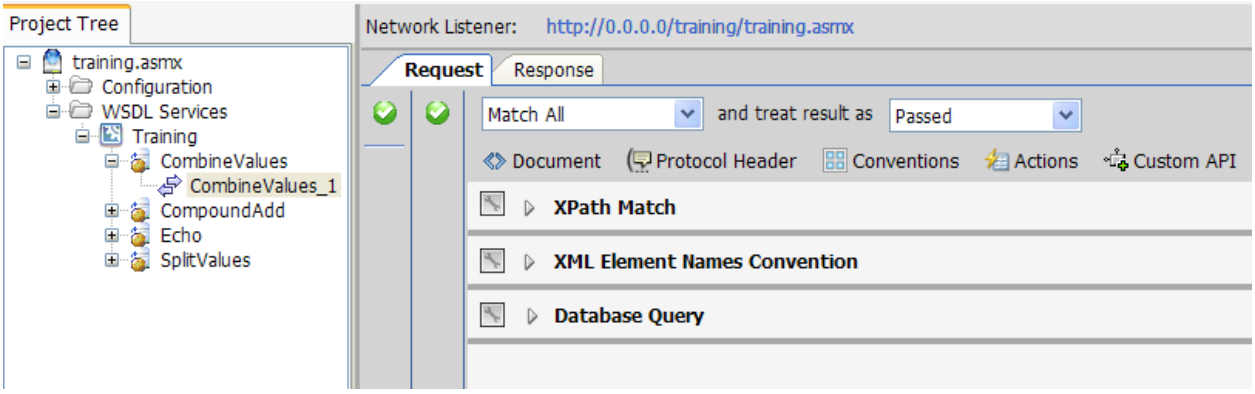
Response Definition

Define simulator response document, including dynamic variable references and task processing. Response document can be imported from file, edited directly, or built using the SOAP generator based on XSD schema.



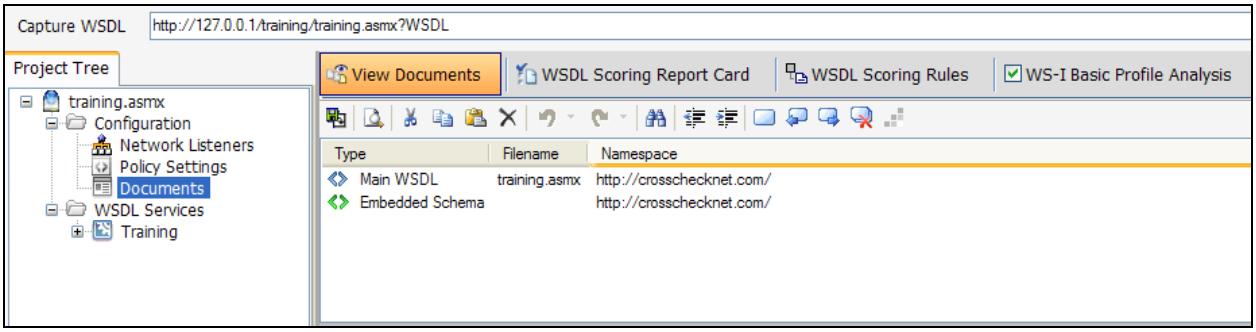
Simulation Settings – Simulator Functional Success Rules

For functional testing, configuration rules for success/failure of inbound messages can be defined from the Success Criteria tab under the request area. This allows a means to generate reports based on functional adherence of the client applications by adhering to rules such as schema format conformance, size guidelines, best-practices naming conventions, etc.



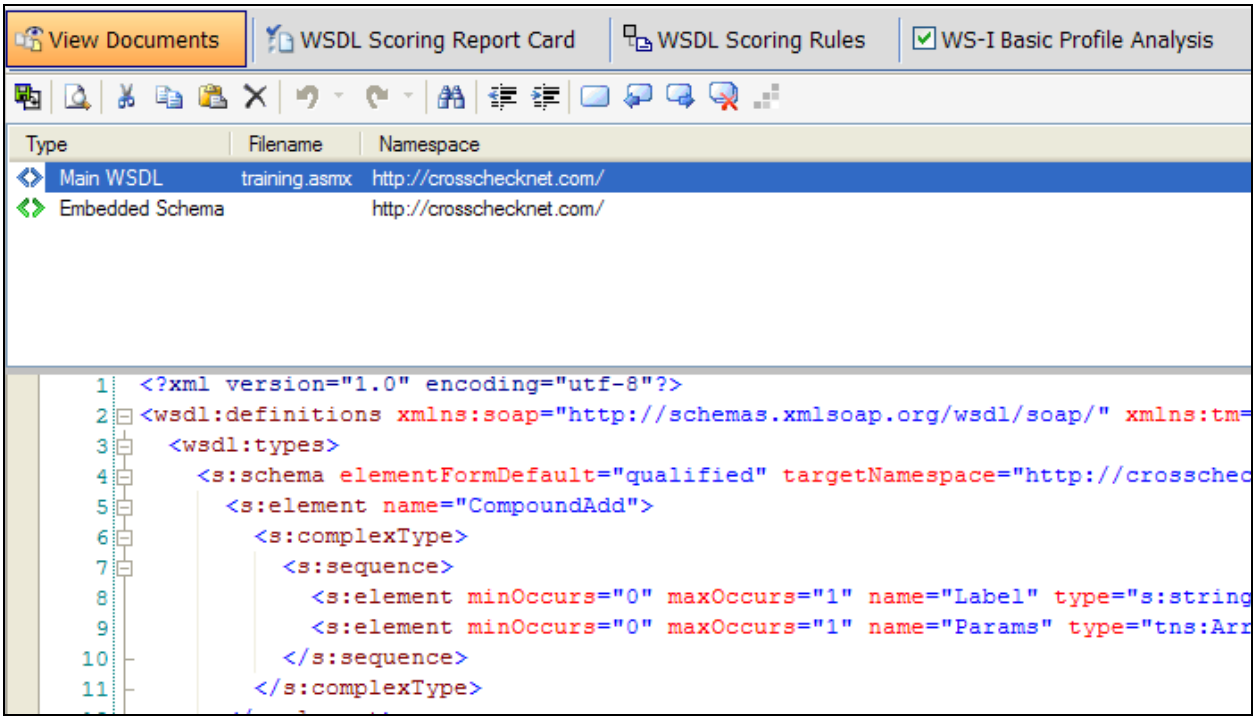
WSDL Analysis

Provides document diagnostics tools to assess and measure the best practices conformance of the WSDL and Schema documents against industry standard specifications as well as custom defined criteria and WSDL scoring.



WSDL Analysis: View Documents

Shows the breakdown of the compound WSDL and schema items for independent analysis and export.

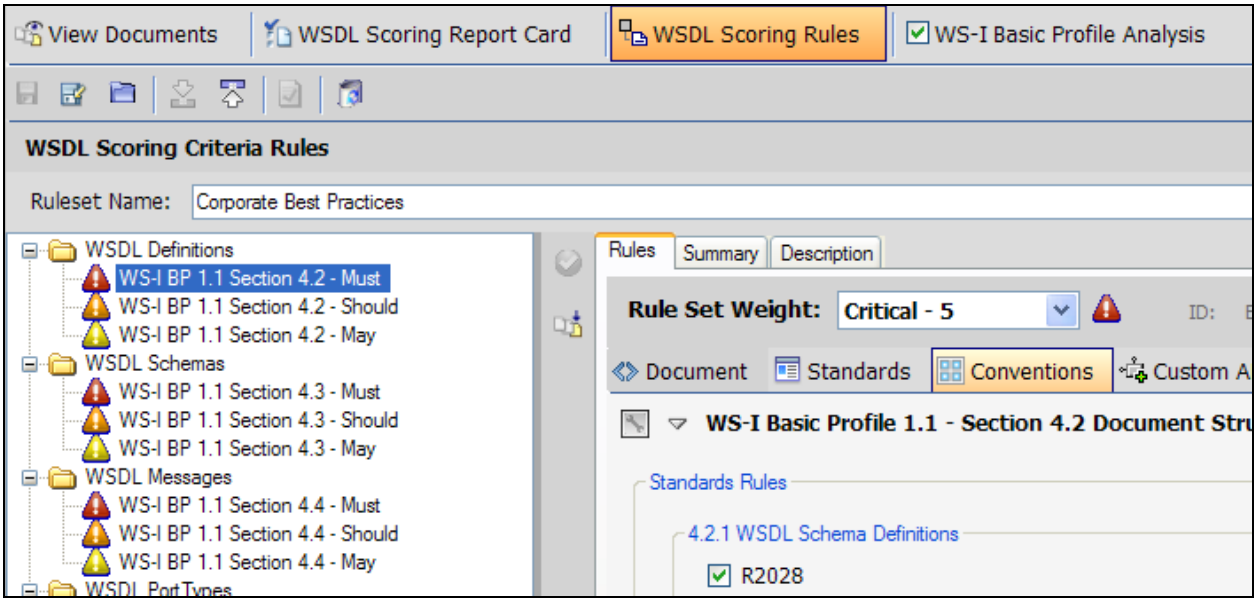


WSDL Analysis: WS-I Basic Profile Analysis

Allows diagnostics against the WSDL and Schema using some of the standard profiles provided by WS-I Basic Profiles.

WSDL Analysis: WSDL Scoring

Provides rules combining industry standard WS-I Basic Profile assertions with custom defined rules to score each component of the WSDL including: WSDL Definitions, WSDL Schemas, WSDL Messages, WSDL Port Types, WSDL Bindings, and WSDL Services.

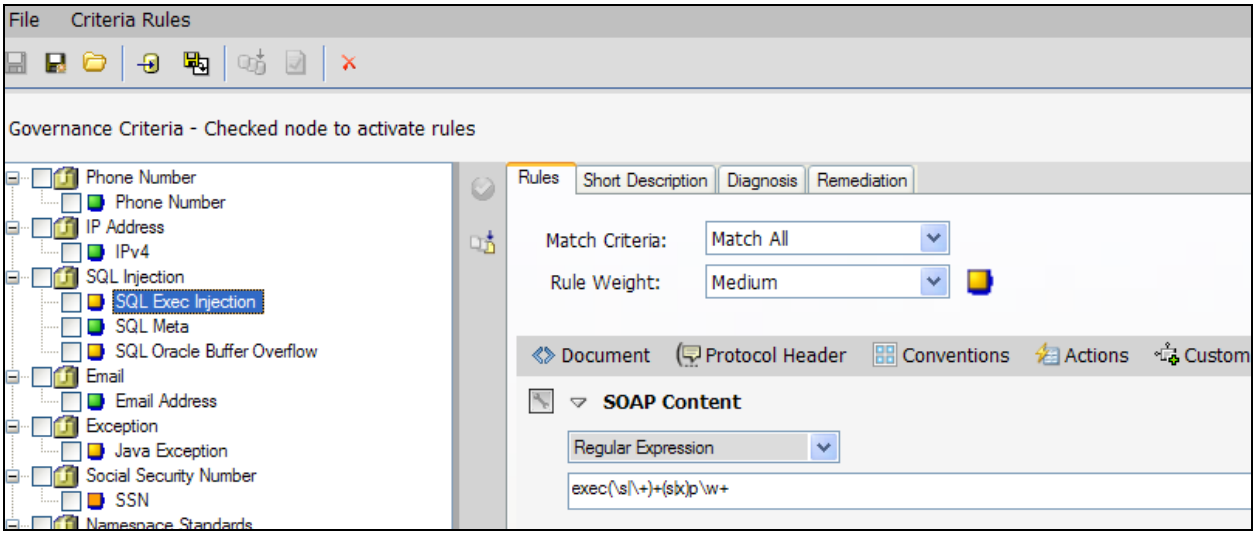


Governance Criteria

Define best practices rules to apply to all inbound document across all running simulations to help insure characteristics of the inbound messages meet the best practices assessment criteria of your organization. Rules are defined with weighting and triggers will be shown when the simulation is running if a governance rule is detected in any inbound document.

Governance Criteria: Rule Set Definition

Define the rule set of best practices characteristics for client requests to running simulators. Governance rules can be selectively enabled or disabled by checking the rule and the governance rule set is portable to other instances of CloudPort to allow for consistent adherence detection to corporate best practices guidelines.

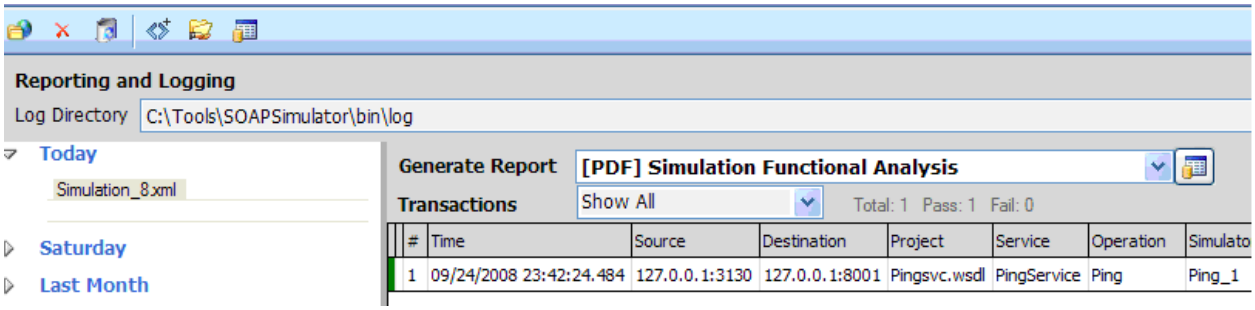


Result Diagnostics

Each simulation run stores all inbound and response results into an XML log file which can be used to inspect and report on various characteristics of the simulation. Reporting option including client functional adherence to success rules, alerts generated from active governance criteria, and raw request/response messages analysis.

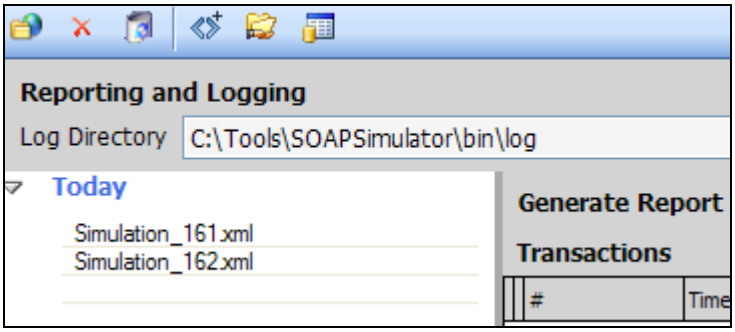
Result Diagnostics: Log View Tree

Shows the log files from simulations. Click on the file to see the contents in the log view on the right and to run reports on the log data.



Result Diagnostics – Simulation Transaction Summary

Shows a summary view of the test results. Summary view will vary depending on the run mode. In QA mode, success results of the test are shown, in performance mode this will show performance statistics for each virtual client and the aggregate statistics, for interoperability and vulnerability modes this will show each dynamic mutated request function. Selecting an item shows the details in the panels below.



Simulation Types

CloudPort allows you to build WSDL based SOAP simulations from a source WSDL and schema, document, or build any arbitrary XML or REST endpoint simulations using custom definitions. All defined simulations are portable and stored in CloudPort project files that can then be run using the runtime CloudPort simulation player.

OpenAPI Simulation Test Case

An OpenAPI Test Case is automatically created based on an OpenAPI 2.0 or 3.0 document. Each OpenAPI operation is parsed and provides the structure of expected format for response. A graphical JSON generator is provided to allow the JSON to be defined in form view or in raw view.

SOAP Simulation Test Case

Point-and-click SOAP simulation is accomplished by simply loading a WSDL. The parsed WSDL and Schema will be represented in the project tab and a simulator for each WSDL operation will be generated automatically based on the schema. Additional settings and additional simulators can then be added for custom business logic simulations. Simulator SOAP responses can be built using the proprietary

Crosscheck Networks SOAP generator which provides graphical representation of the schema for the message where the SOAP document is subsequently generated automatically.

XML Simulation Test Case

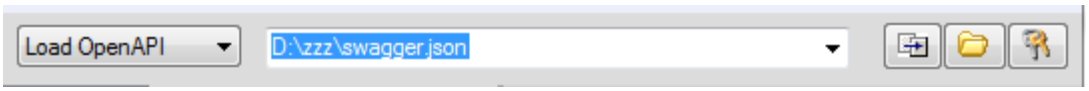
An XML Simulation provides the means to import or create any custom request/response transaction sequence. You can also combine XML Simulators with SOAP Simulators.

REST Simulation Test Case

A REST Simulation provides the means to define input headers to target the simulation rules.

OpenAPI Capture and Processing

When testing APIs, an OpenAPI 2.0 Swagger or OpenAPI 3.0 YAML document may be available that provides the contract of information about the expected structure of each request and response for the service to allow clients to invoke simulated API responses. CloudPort parses OpenAPI 2.0 and 3.0 documents for API simulation and builds the response templates for all defined operations. When an OpenAPI document is captured, it becomes a new Project item in the project tree with various configuration options for the test environment and creating simulation rules for each API operation.



You can capture an OpenAPI document using several supported methods, including

- Drag-Drop – Drag and drop OpenAPI files from a file explorer directly into the project window
- File – Browse your file system for the OpenAPI document to load
- HTTP – Fetch the OpenAPI document using HTTP
- HTTPS – Fetch the OpenAPI document using HTTPS
- HTTP Basic Auth – Fetch the OpenAPI document using HTTP with Basic Authentication Credentials
- HTTPS Basic Auth - Fetch the OpenAPI document using HTTPS with Basic Authentication Credentials
- HTTPS X.509 Auth - Fetch the OpenAPI document using HTTPS and X.509 SSL Authentication
- Project – Load OpenAPI document previously saved to a project file

For http or https access, authentication options are provided using the “Authentication Options” button at the right of the OpenAPI Location field. Both HTTP Basic Authentication and SSL X.509 client authentication are supported. SSL X.509 client authentication is supported through a fully integrated native PKI management interface that allows you direct access to your windows certificates and private keys. Simply select the certificate you want to use for SSL client cert from the list of X509s loaded on your local machine.

Once the OpenAPI document is captured and parsed, the project tree will show a graphical representation of the OpenAPI contents. Each node in the project tree will show properties specific to the type of object (i.e. Service Tag, OpenAPI Operation, Test Case). When you select an item in the tree, the available options for that item will appear on the right portion of the screen.

WSDL Capture and Processing

To create a Web Services Simulation, a WSDL document provides the contract of information about the expected structure of each request and response for the service to allow clients to build requests to consume the services responses. CloudPort parses WSDL documents for service emulation and exposes the new simulated service WSDL and schema on a new network listener port. When a WSDL is captured, it becomes a new Project item in the project tree with various configuration options for the simulation environment and creating one or more request/response simulators for each service operation.

Capturing a WSDL document will automatically build SOAP Simulators based on the settings enabled in the File->Settings panel. You can capture a WSDL document using several supported methods, including

- Drag-Drop – Drag and drop WSDL files from a file explorer directly into the project window
- File – Browse your file system for the WSDL to load
- HTTP – Fetch the WSDL using HTTP
- HTTPS – Fetch the WSDL using HTTPS
- HTTP Basic Auth – Fetch the WSDL using HTTP with Basic Authentication Credentials
- HTTPS Basic Auth - Fetch the WSDLs using HTTPS with Basic Authentication Credentials
- HTTPS X.509 Auth - Fetch the WSDLs using HTTPS and X.509 SSL Authentication
- Project – Load WSDLs previously saved to a project file

For http or https access, authentication options are provided using the “Authentication Options” button at the right of the WSDL Location field. CloudPort supports both HTTP Basic Authentication and SSL X.509 client authentication. SSL X.509 client authentication is supported through a fully integrated native PKI management interface that allows you direct access to your windows certificates and private keys. Simply select the certificate you want to use for SSL client cert from the list of X509s loaded on your local machine.

Once the WSDL is captured and parsed, the WSDL project tree will show a graphical representation of the WSDL contents. Each node in the WSDL project tree will show properties specific to the type of object (i.e. Service, Operation, Simulator). When you select an item in the tree, the available options for that item will appear on the right portion of the screen.

Dynamic Generated CloudPort WSDLs

Once a WSDL and associated Schema is loaded and a simulation is running, the WSDL and Schema can be accesses directly from the configured listener via a HTTP GET request from a web browser or any web services capable client.

For example, if the Web Service simulation is configured on the following URI:

http://10.5.3.1:80/myservices/myservice.asmx

then the new simulated WSDL and schema can be accessed by adding ?WSDL as follows:

http://10.5.3.1:80/myservices/myservice.asmx?WSDL

Working with Projects

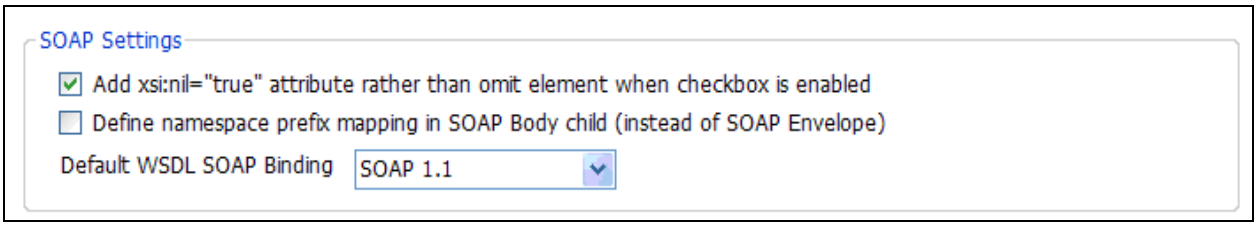
All the settings configured in the Simulation Settings are preserved when you save a CloudPort project file. Project files can be shared among team members, checked into version control to synchronize with service release versions, and are used by the command-line interface. Project management also allows subprojects to be combined into master projects and merging awareness will allow merging of projects and subprojects.

Default Project Settings and Behavior

Various configuration settings are provided to allow customization of the SOAP generation and WSDL parsing behavior. To configure these settings, go to the File->Settings dialog and choose the settings for how Simulators get generated and for the generated SOAP messages whether to automatically fill field entries. For more information on each setting, go to the project settings section.

WSDL Parsing and SOAP Generator Settings

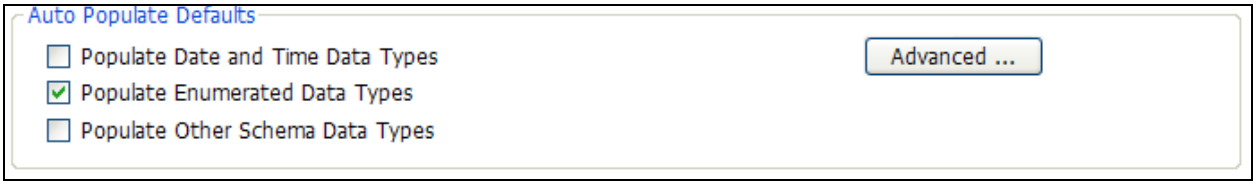
SOAP Settings



Add xsi:nil=”true” attribute rather than omit element when checkbox is enabled
With this option enabled, a checkbox placed in front of an element in the fields view will result in the generated SOAP message having an xsi:nil=”true” attribute added to the element node (i.e. <test xsi:nil=”true”/>. When this option is disabled (default), a checkbox in front of the element in Schema Fields view will result in the element being omitted from the generated SOAP request. The xsi:nil=”true” setting is required by some servers which expect SOAP encoded requests to preserve the XML structure with nil contents explicitly specified.

Define namespace prefix mapping in SOAP Body child (instead of SOAP Envelope)
This option allows you to choose where the document prefix namespace mappings are defined. While both options produce semantically equivalent documents, some non XML-compliant back-end servers do not properly recognize namespaces defined in the SOAP Envelope and expect them rather to be defined within the first SOAP Body child element. Checking this option will result in the generated SOAP messages to define the namespaces in the SOAP Body child. The default unchecked behavior will create the prefix namespace references in the SOAP element.

Auto populate Defaults



Populate Date and Time Data Types
Auto-detect parameter items which have a date or time data type and populate the value with a date/time value representing the current date and time.

Populate Enumerated Data Types

Auto-detect parameter items which have enumeration facets and populate the values with the first enumerated item in the referenced items.

Populate Other Data
Auto-detect the type based on the XSD schema and provides a schema compliant value representative of the referenced data type.

Advanced Settings
The advanced settings provide some additional automatic data populate options. For optimal performance when parsing WSDLs and working with Simulators, it is recommended to keep these settings as their defaults.

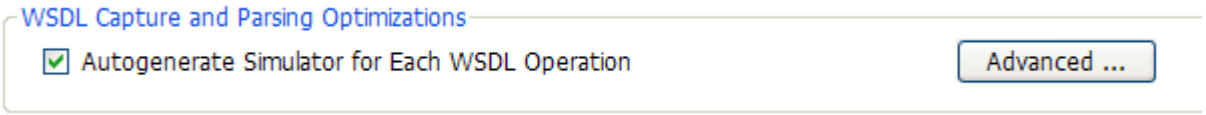
Allocate Arrays
Auto-detect array items represented as schema objects of complex or simple type with attribute of maxOccurs > 0 and include these object in the default build SOAP request. When dealing with complex schemas, it is recommended to keep this setting unchecked and manually from the Schema Fields view enable each instance of an array to be included in the request.

Enable Attributes
Auto-detect attribute values and enable all of them. When dealing with complex schemas, it is recommended to keep this setting unchecked and manually from the Schema Fields view enable each attribute to be included in the request.

Enable Optional Elements
Auto-detect SOAP elements defined with XSD schema attribute minOccurs="0" mean that they are optional elements and may not need to be present in the SOAP request. This setting allows you to enable all instances of optional elements such that they all appear in the generated SOAP. When dealing with complex schemas, it is recommended to keep this setting unchecked and manually from the Schema Fields view enable each SOAP element to be included in the request.

WSDL Capture and Parsing Optimizations

The settings within this section need only be altered if you are experiencing excessively long delays in parsing the WSDL and rendering the graphical items in the Schema Fields view. Delays usually indicate that there are complex recursive schema definitions causing the number of permutations for test configurations becomes CPU intensive for CloudPort to represent graphically. The optimization settings will detect these hierarchical schema declarations and limit the depth of recursive processing.



Auto generate Simulator for Each WSDL Operation
To improve speed in creating new WSDL projects, disable the automatic Simulator generation to allow you to manually create each Simulator from the selected WSDL operation

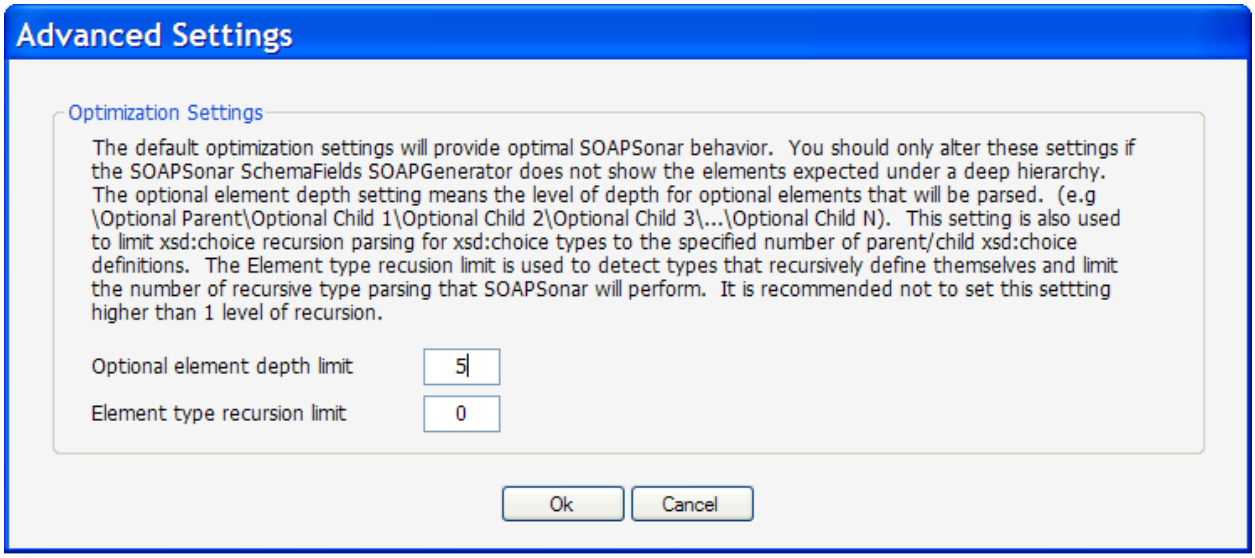
Advanced Settings
The advanced settings allow you to change the default optimizations for CloudPort. For optimal performance when parsing WSDLs and working with Simulators, it is recommended to keep these settings as their defaults.

Optimize Schema Optional Element Parsing Depth

Optional Elements are elements defined as minOccurs=0, or having a nillable=true attribute. These elements do not need to be included in the SOAP request. This optimization depth parsing setting will limit the recursive depth of optional elements with optional children elements to a limited depth of recursion (default = 5). Alter this setting if you do not see the elements you expect to see in Schema Fields view.

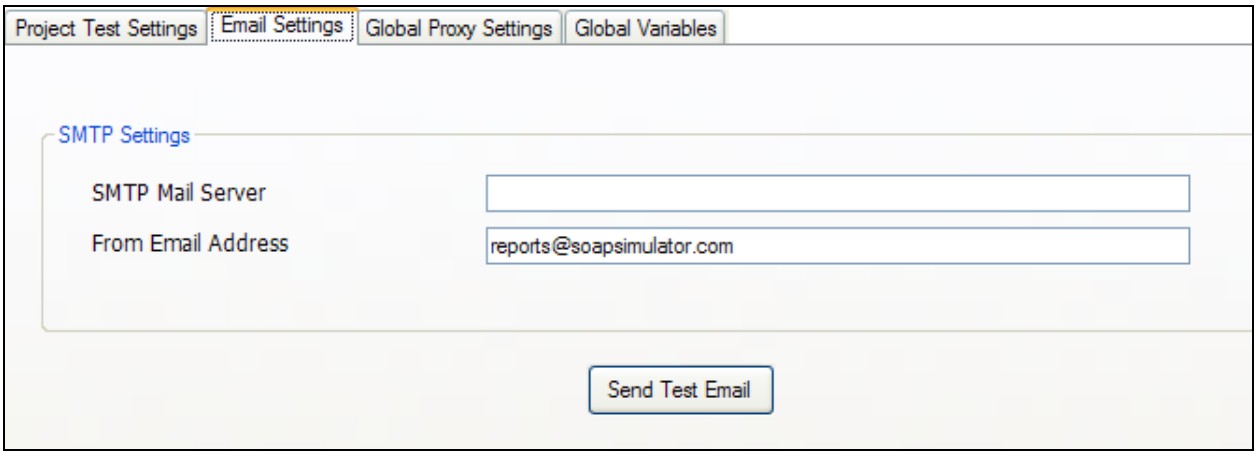
Optimize Recursive Schema Definition Parsing Depth

Sometime schema elements are defined recursively to include themselves. This is poor practice for schema authoring and can cause significant overhead in repeatedly parsing the same type declarations over and over. This setting allows recursive parsing to the specified depth. The default setting is 0 to prevent recursive type parsing. Alter this setting if you see the elements you expect to see in Schema Fields view, but it is only a text field, and not a complex type.



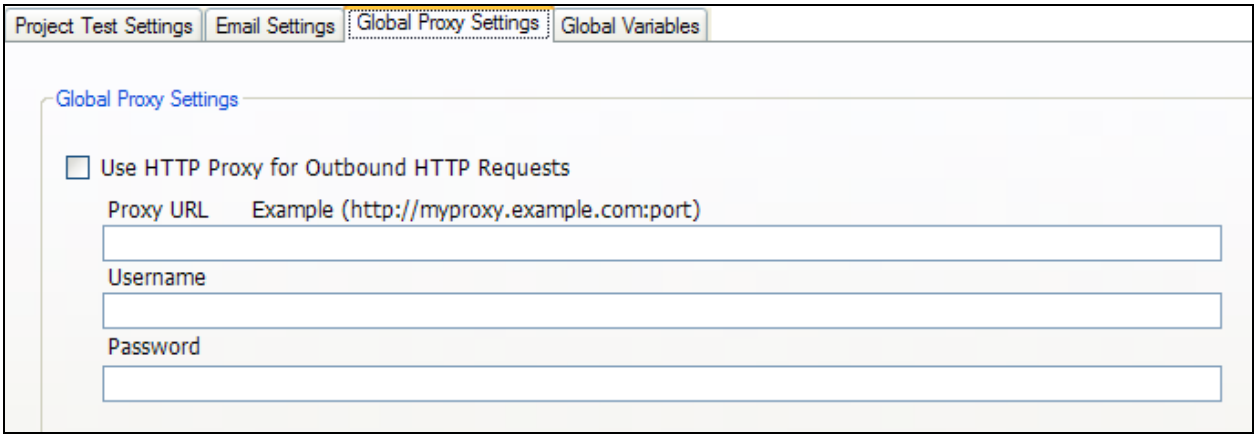
Email Settings

Settings for the SMTP server and from email address that are used by the command-line interface when the email report option has been enabled.



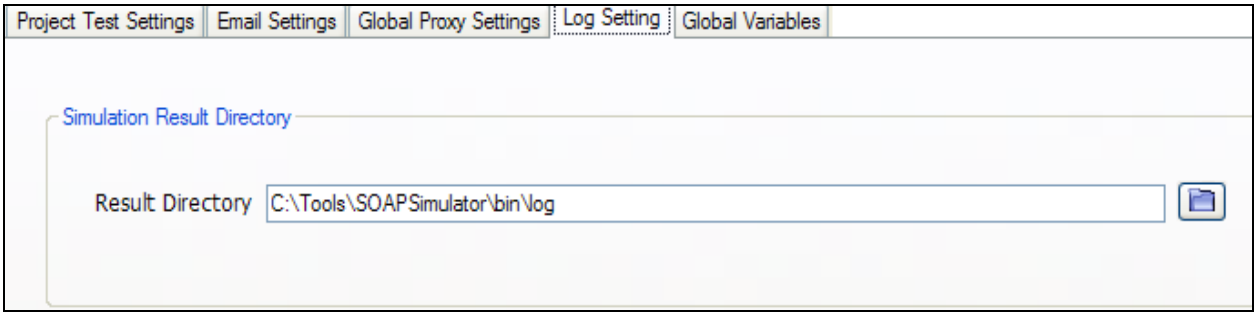
Global Proxy Settings

Enables the use of a proxy server for WSDL capture events when you use CloudPort to access a WSDL via HTTP or HTTPS that requires a proxy server to gain access..



Default Log Directory Setting

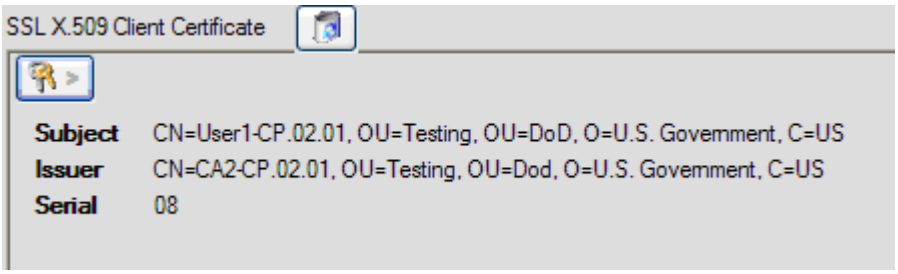
Configure the default location to write results from running simulations.



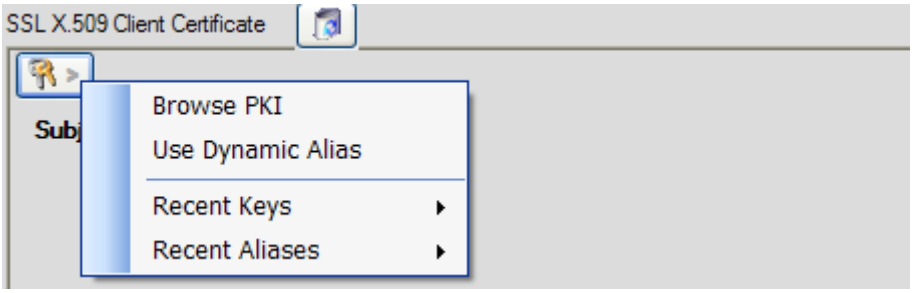
X509 Key Selection

Selecting keying information can be done explicitly from a Windows Keystore, Java Keystore, File, or Smartcard. Another feature is known as Key Aliasing which allows an alias placeholder to be defined instead of an actual certificate. Key aliases are resolved dynamically at test time when running a test in project view, or running a test suite in run view.

Selecting a Certificate



Define Dynamic Certificate Alias



BUILDING SERVICE SIMULATIONS

Crosscheck service emulation allows you to build end-point server simulations for content such as HTML, XML, JSON, EDIT, and SOAP. Emulations can be built manually across different types of message formats, or imported from captured traffic, which also includes the ability to automatically emulate the timing characteristics of the traffic pattern.

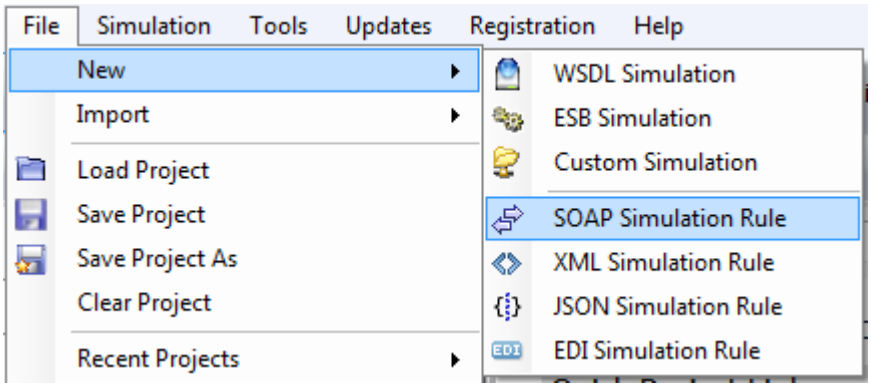
Subtopics in this section include

Simulation Protocols – HTTP, MQ, EMS, WLS, and JMS

Service simulations can be defined and run for HTTP, HTTPs, IBM MQ, Tibco EMS, Weblogic JMS, and other native JMS protocols. For HTTP and HTTPs end-point simulation, CloudPort listens directly on the network for inbound request and processes then requests and provides the simulated responses according to the defined rules. For native MQ and native JMS simulations, CloudPort will take the place of the back-end message processing system by reading from the inbound queue, processing the message, and writing to the processed queue. This allows clients to send messages to the designated queue without requiring the back-end message processing server to be processing the messages. CloudPort will process the messages under the same policy rules that apply to direct HTTP or HTTPs end-point simulation traffic, including task processing, success criteria evaluation, security and identity processing.

SOAP Simulation Rules

Point-and-click service simulation is accomplished automatically based on a WSDL and the parsed schema. Each WSDL operation provides the structure of expected format to identify the inbound request and map it to a simulated response. The Crosscheck Networks SOAP generator provides an intuitive graphical representation of the document schema to define the fields and structure which result in automatic SOAP document generation.



To create a SOAP Simulators from a WSDL document, first go to the Capture WSDL area, then choose a file or network location where the WSDL resides. Once a WSDL document is captured and parsed, the project tree will show a set of nodes representing the parsed WSDL and providing access to global test settings and Simulator generation.

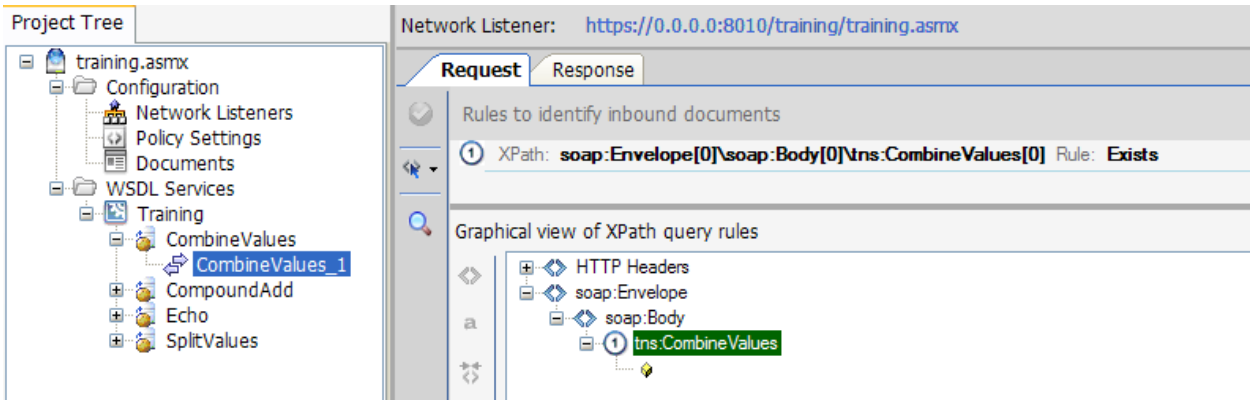
By default a Simulator will be automatically created for each WSDL operation. The Simulator node can be accessed by navigating to the WSDL operation node and expanding the contents. Click on the Simulator node to access the Request/Response Simulation configuration panels.

Simulator Configuration Components:

Request Processing
Define inbound document identification, processing, and success rules. Sample documents can be imported from file, edited directly, or built using the SOAP generator based on XSD schema.

Response Definition

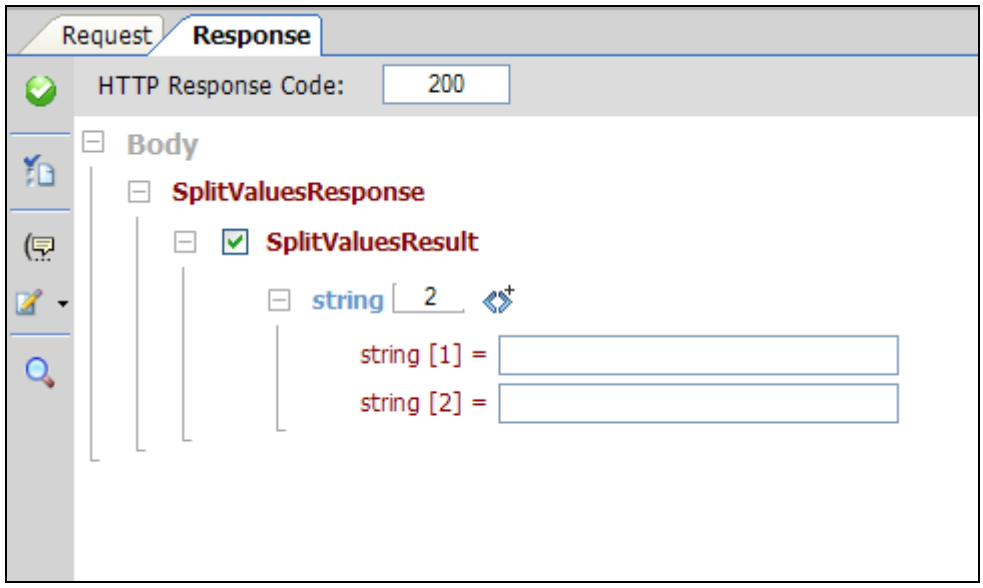
Define simulator response document, including dynamic variable references and task processing. Response document can be imported from file, edited directly, or built using the SOAP generator based on XSD schema.



WSDL SOAP Simulators have a graphical SOAP generator which is visible from the Schema Fields tab. To create new SOAP Simulators, right-click the WSDL operation node, or use the File->New->SOAP Simulator menu item. In order for the Schema Fields SOAP generator to know the appropriate schema to use for the request generation, the WSDL operation to invoke must be selected when creating the SOAP Simulator type.

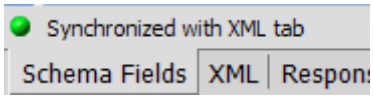
Schema Fields SOAP Generator

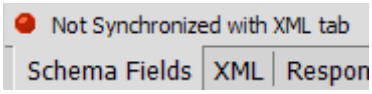
The Schema Fields tab provides a graphical set of objects representing each schema element applicable for the SOAP request for that WSDL operation. Each type of schema declaration has a corresponding GUI widget to configure it. The data provided in the fields will be used to generate the SOAP Simulator response for the current Simulator.



GUI widgets that dynamically appear will be based on the data type of the schema element that is being represented. Each type of data entry widget is explained below.

For custom XML or custom SOAP, the green icon labeled **Synchronized with XML tab** just above the Schema View tab allows the Schema Field SOAP generator to be disabled. When this icon is green, the XML tab will show the resulting generated SOAP message based on the schema mappings defined. When this icon is red, the SOAP generator is disabled and data can be directly imported or defined on the XML tab.



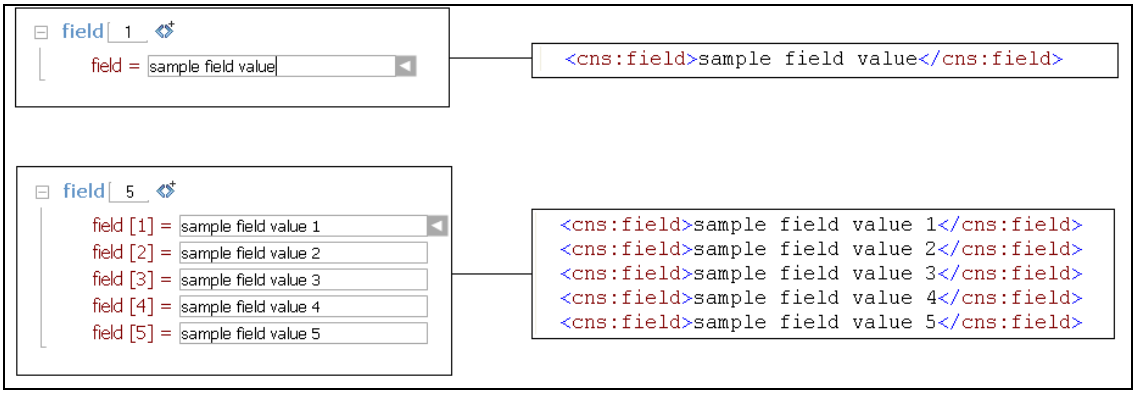


Features of Schema Fields SOAP Generator

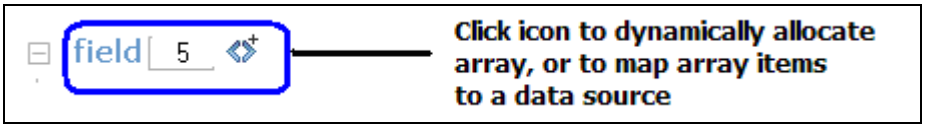
In addition to providing easy parameter entry for SOAP values, the Schema Fields view also provides a rich set of features that simplify creation of Simulators and test data entry editing. The types of GUI items that appear based on the underlying schema type are described below.

Data Entry: Dynamic Array Structures

For XSD elements defined with maxOccurs="N" and N > 0, a dynamic array boundary graphic will appear for this array object to allow you to choose the array boundary (i.e. the number of occurrences to generate for this element and its child elements)

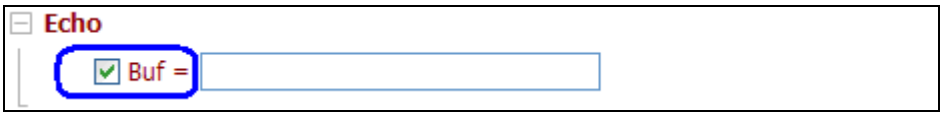


For example, the above setting would generate 1 instance of the detail level element. Setting this value to 5 would generate 5 instances of the detail level element, each new instance having a separately defined parameter.



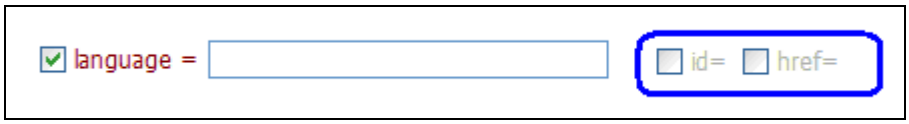
Data Entry: Optional Elements

For XSD elements defined with minOccurs="0" or with a nillable attribute value of true this means that the element may be omitted from the message. For these elements, a checkbox will be visible to include or omit the element when generating the SOAP request.



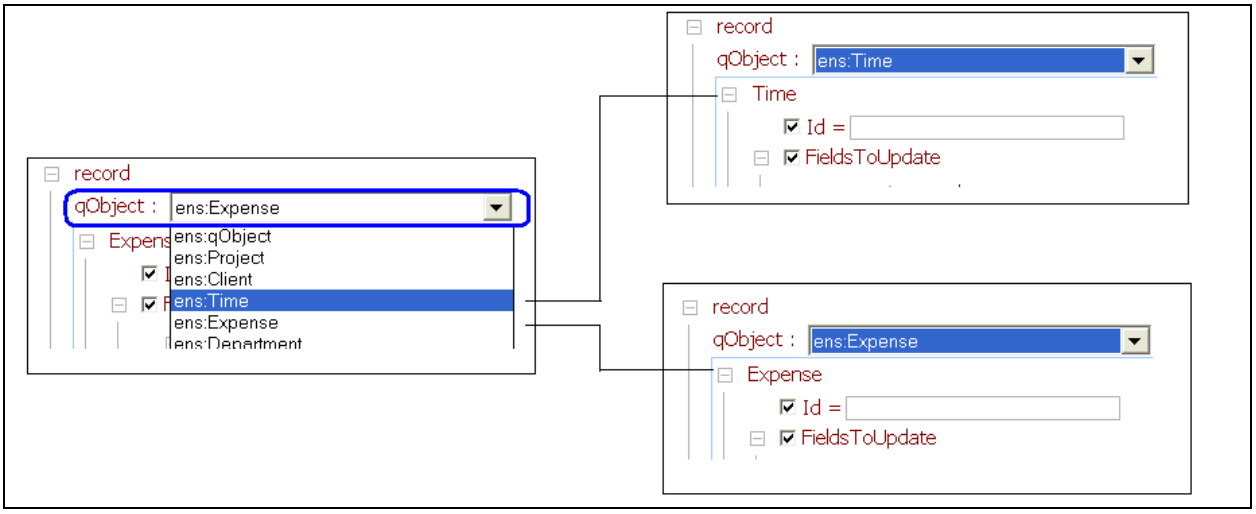
Data Entry: Optional Attributes

Attributes appear with checkboxes so that you can omit or include the attribute when generating the SOAP request.



Data Entry: Abstract Types (XSD Polymorphism)

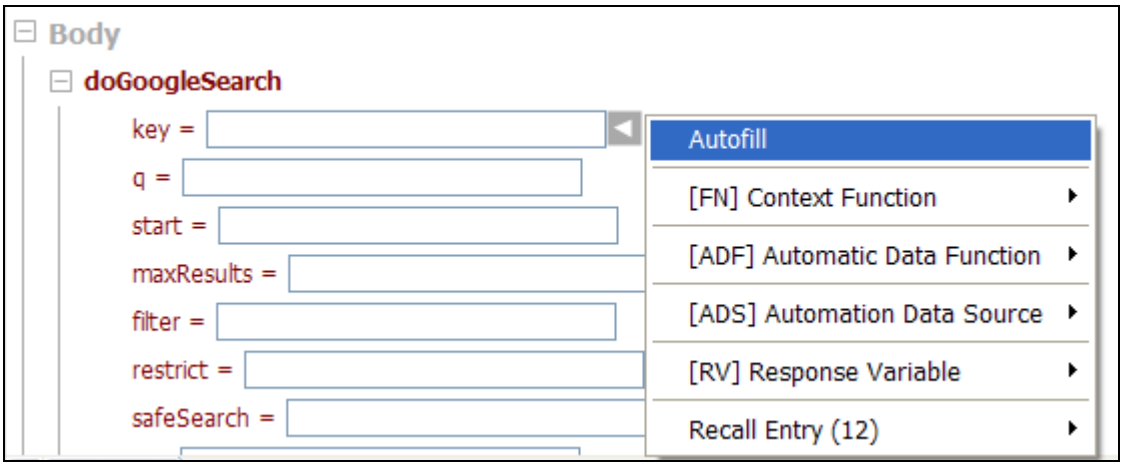
For XSD elements defined as abstract=true, where other XSD elements extend the base abstract type, this correlates to the notion of late-binding (the binding of a data-type to an object after the object has been instantiated). Each extended type based on the abstract type will appear in a pull-down list to dynamically alter the binding of the extended type to use when generating the SOAP request:



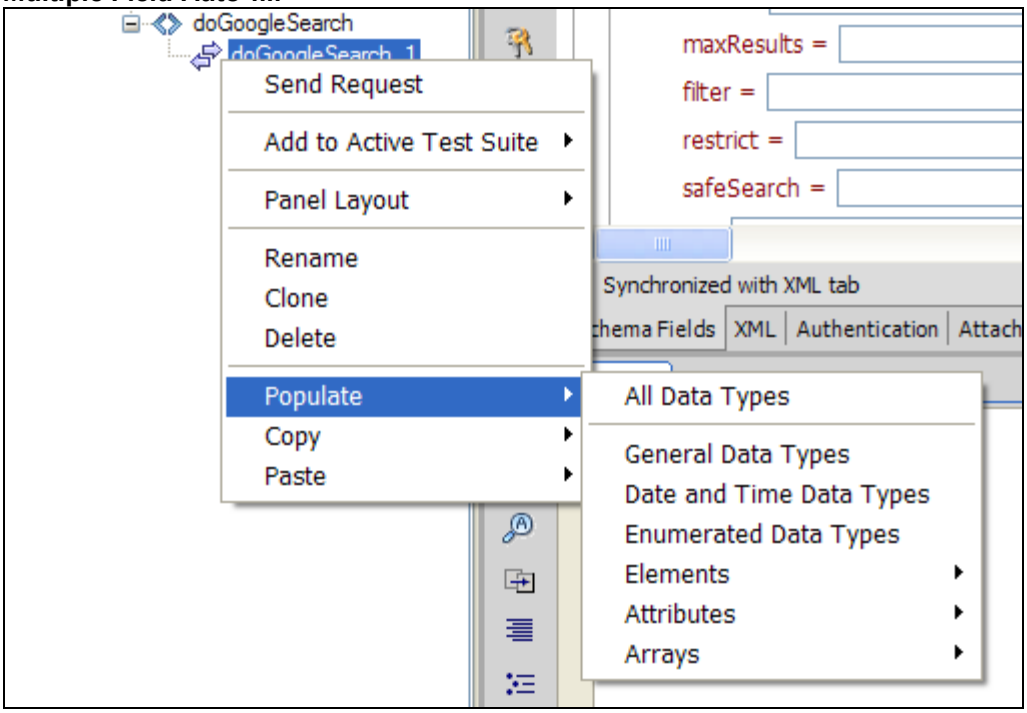
Data Entry: Auto fill

To generate a data value automatically which is schema compliant based on the XSD data type, select Auto fill from the right field option menu. Each generated value auto-detects the schema type. To automatically fill all data type values, right-click on the Simulator node in the project tree and select the Populate option.

Single Field Auto-fill

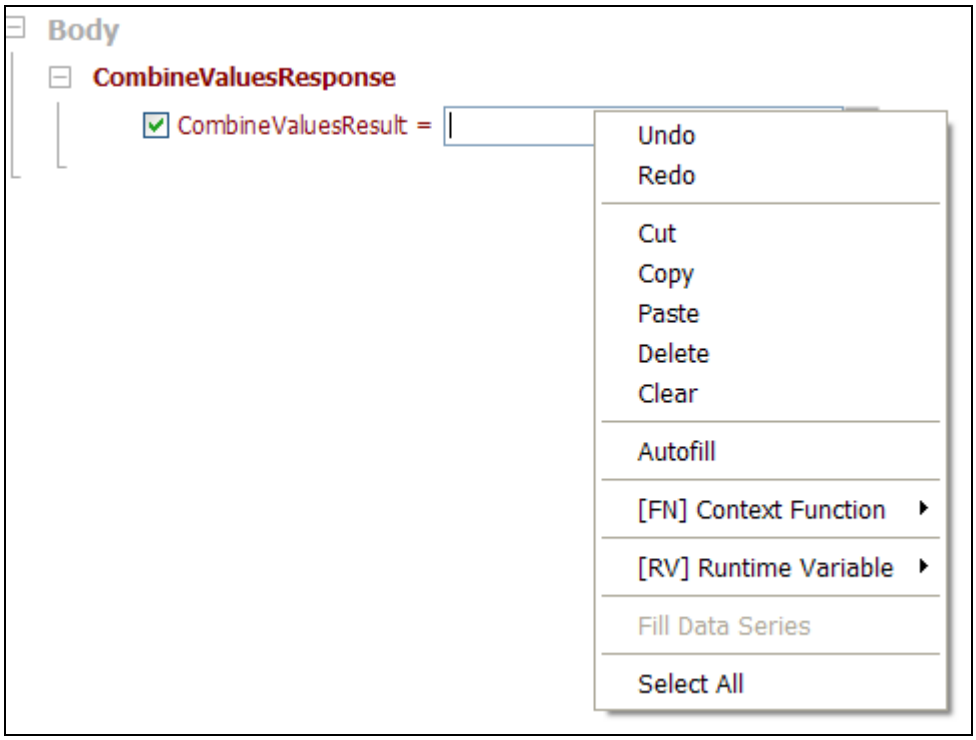


Multiple Field Auto-fill



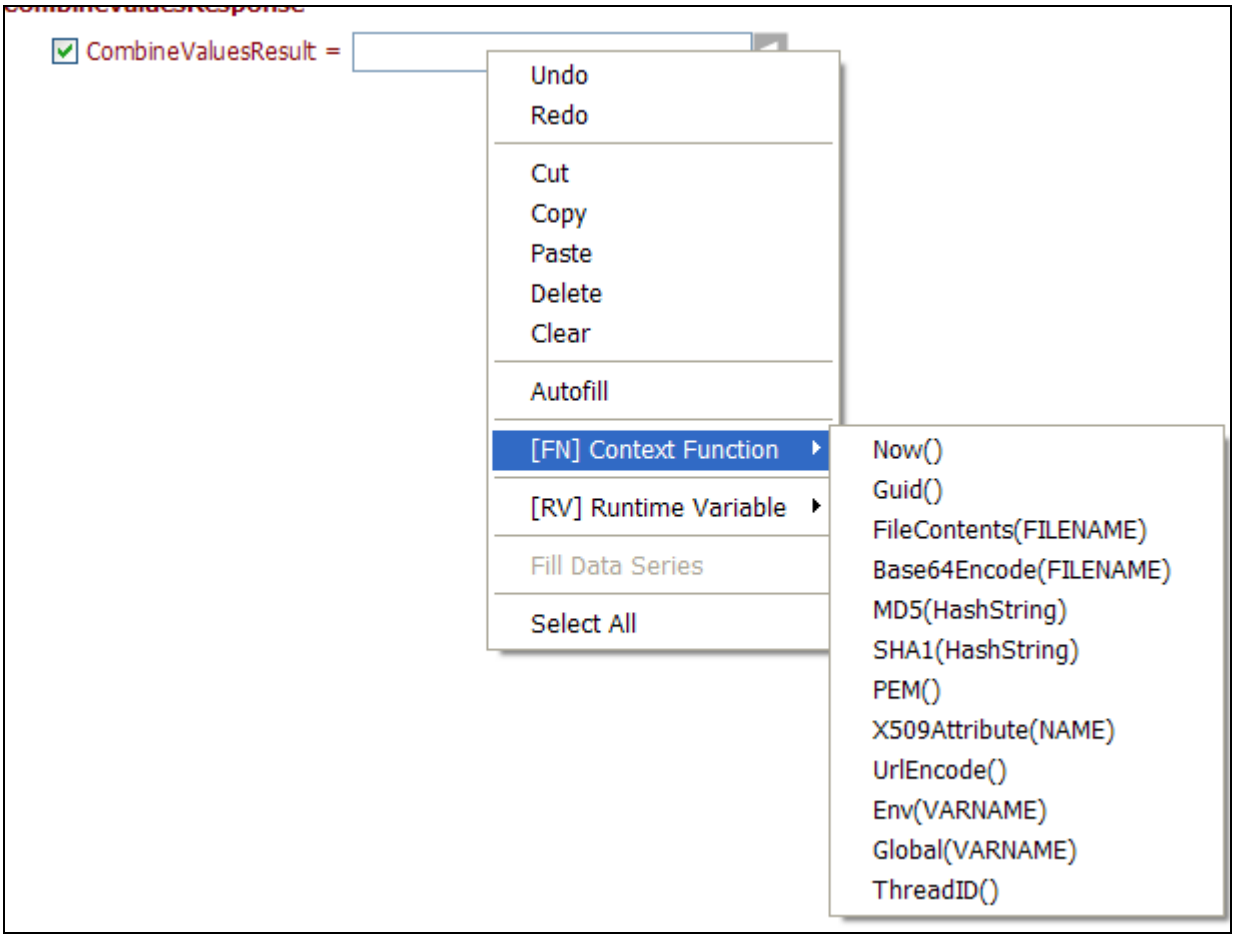
Data Entry: Variable Parameters

Request values can be defined statically, or defined as variables. Variable options include data source driven testing, data inputs created via ADF libraries, or response values from dependent Simulators. To choose a variable parameter, right-click on the Simulator node in the project tree and select the type of variable to create.



Data Entry: Context Functions

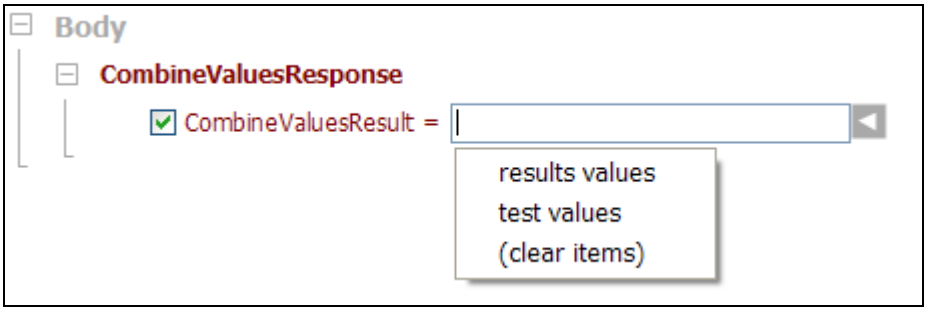
Context Functions provide inline data processing with value replacement occurring each time the request is sent. (Context functions and parameters are discussed in more detail in the Context Function section).



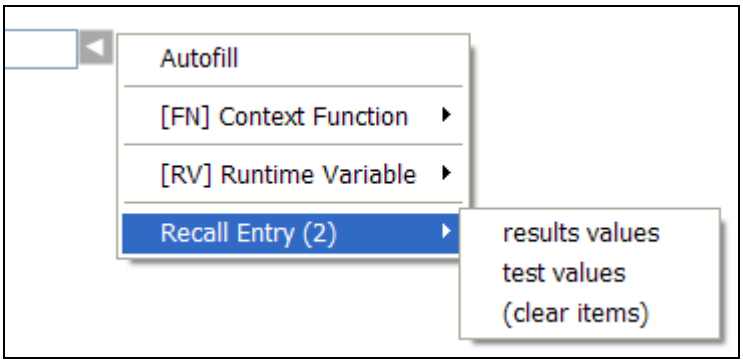
Data Entry: Entry Recall

A handy editing feature which his commonly seen in web browsers is also provided by CloudPort . The entry recall tracking will recall all values previously entered into this parameter field. Value are stored and recalled based on each WSDL project loaded.

This feature can be invoked by pressing the <down-arrow> key while the cursor is in the parameter field



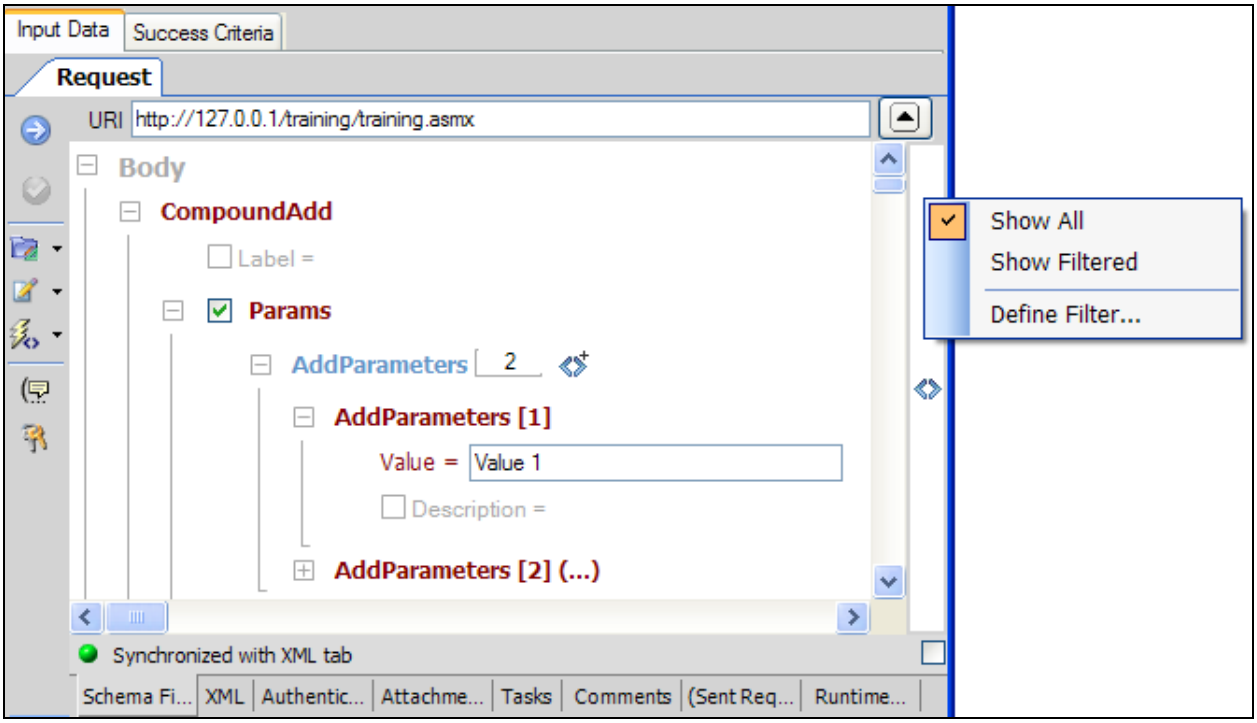
You can also use the context menu shown at the right of the control to select the “Recall Entry” option and select from the list of previously entered items.



Schema Fields View Filtering

By default all schema items related to the selected WSDL operation will be displayed for allocation when using the SOAP generator. This allow you to build any variation of SOAP as defined by the schema of the WSDL. However, often in practice only a subset of these schema items are of interest to view and allocate.

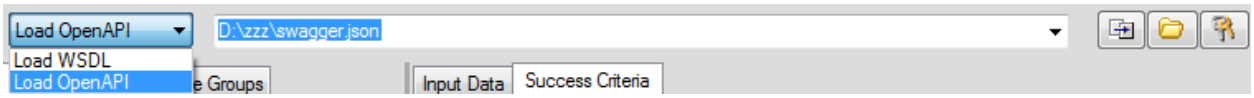
You can choose to hide fields that you do not want to allocate and filter the display to only show the fields of interest to allocate values for the generated SOAP. To access the filtering options, click on the menu bar to the right of the schema fields area. Options include “Show All”, “Show Filtered”, and “Define Filter…”. Choose Show All to view all schema field items, choose Show Filtered to filter the schema items to those defined the last time the “Define Filter” option was selected and the filter was defined.



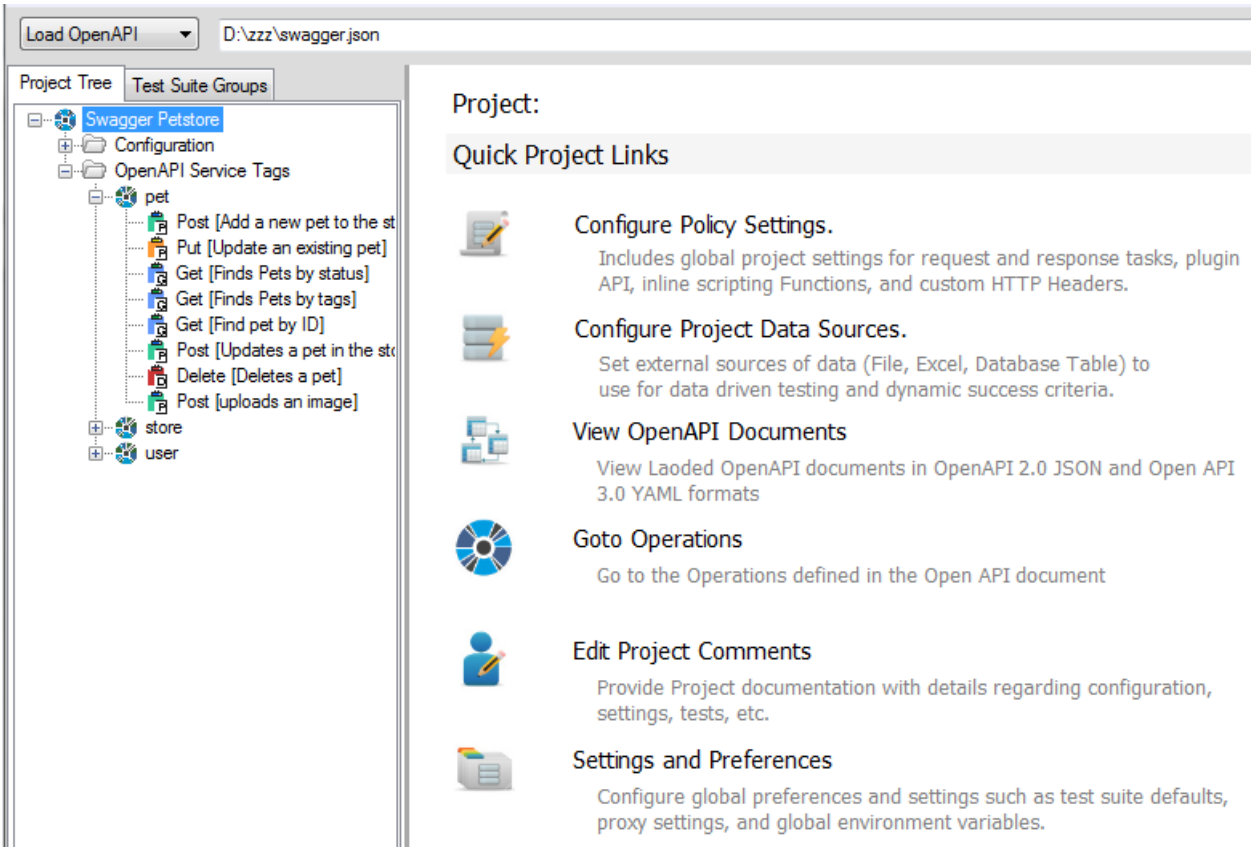
OpenAPI Simulation Test Case

An OpenAPI Simulation Test Case is automatically created based on a loaded OpenAPI 2.0 or 3.0 Swagger or YAML document. The OpenAPI operations are parsed and extracted from the document and the project will show all defined operations and enable OpenAPI test cases to be generated using the underlying OpenAPI definitions as the template. A graphical schema fields view enables a Form based generator to build the JSON response, or the JSON response can be edited in Raw format from the JSON tab.

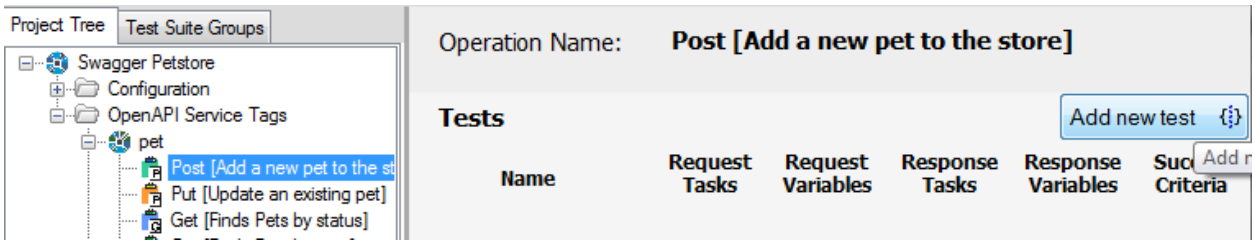
To create OpenAPI test cases from an OpenAPI document, first go to the top left of the Project view screen and select “Load OpenAPI” from the Load selection and then choose a file or network location where the OpenAPI document resides. Once an OpenAPI document is captured and parsed, the project tree will show a set of nodes representing the parsed OpenAPI document and providing access to global test settings and test case generation.



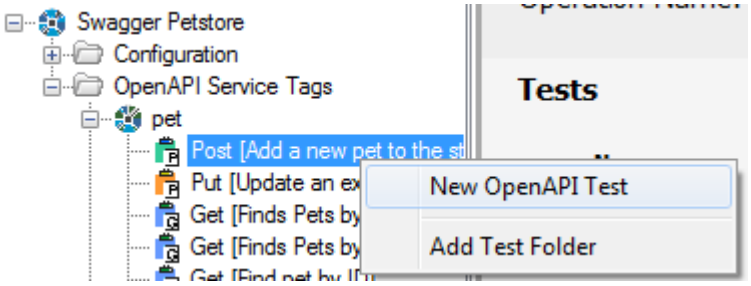
Each Service Tag and Operation defined within the OpenAPI document will be parsed and shown in tree format.



To create a new OpenAPI test case, simply click on the operation and choose “Add New Test” button



or right-click on the OpenAPI operation and choose



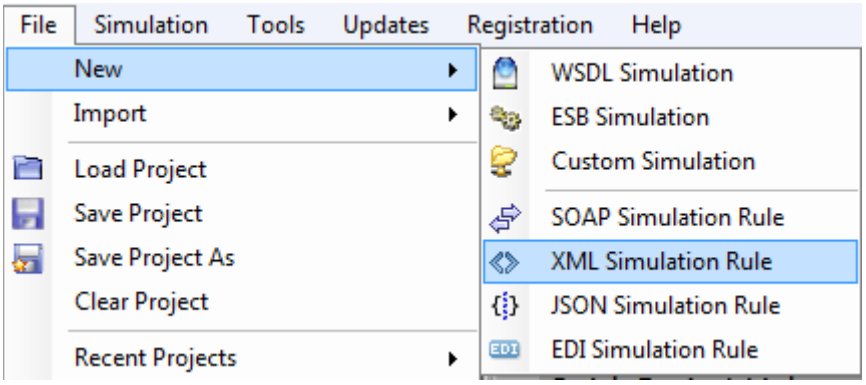
Schema Fields Generator

The JSON message format can be generated using the Schema Fields view or via the JSON tab to see the raw message view. The schema fields generator works the same for OpenAPI JSON test case messages as it does for WSDL-based SOAP messages. Refer to the Schema Fields Generator section under SOAP test case type for more details about the Schema Fields message generator.

XML Simulation Rules

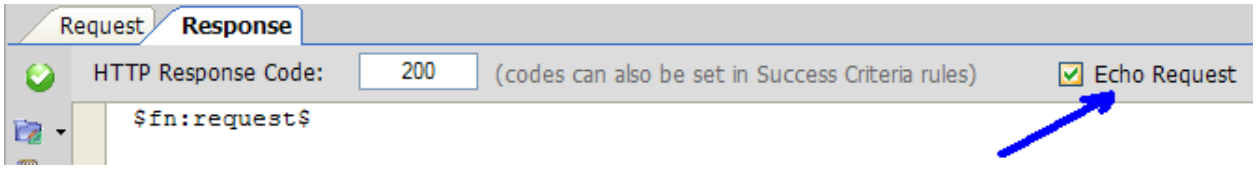
An XML Simulation provides the means to create any arbitrary request/response simulation. This Simulation rule type can be created under a WSDL Operation or under a Custom Test Group.

To create an XML simulation for a WSDL project, navigate to an existing WSDL project operation node and select File->New->XML Simulator. To create an XML Simulator for any endpoint, create or select a Custom Test Group, then select File->New->XML Simulator. The XML Simulator type allows you to load the request data from file, paste from clipboard, or type in the request data manually. Variable parameters are supported anywhere within the XML data section and the HTTP Header for ADS, RV, and Context variables.



Create an Echo Reflection Service

To set up a simulator test that uses the request as the response, click on the “Echo Request” checkbox on the request tab, or just enter the context function “$fn:request$” directly in the response area.

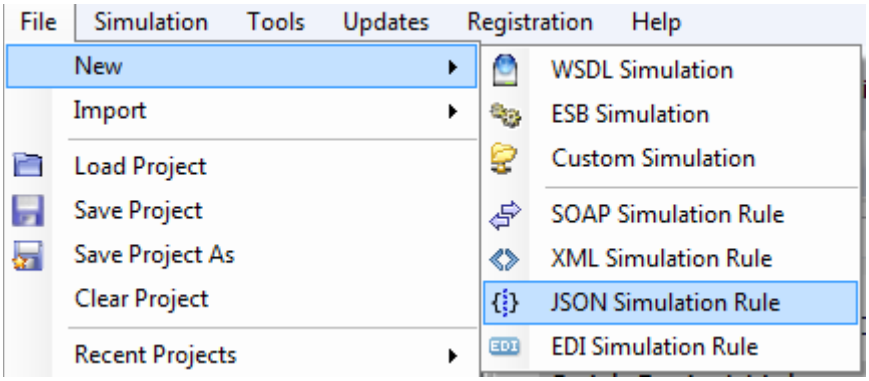


An echo service is also packaged by default and available for selection when launching the CloudPort runtime simulation player.

JSON Simulation Rules

A JSON simulation provides the means to create any arbitrary request/response simulation with JSON message formats. This Simulation rule type can be created under a WSDL Operation or under a Custom Test Group.

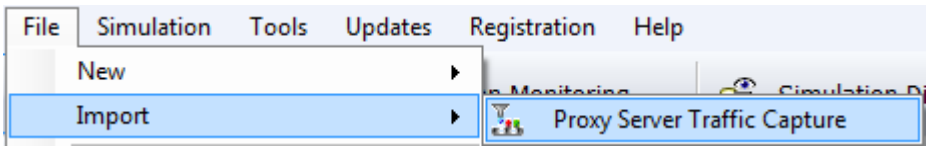
To create a JSON simulation select File->New->JSON Simulator. The JSON Simulator type allows you to load the request data from file, paste from clipboard, or type in the request data manually. Variable parameters are supported anywhere within the JSON data section and the HTTP Header for data source, RV, and Context variables.



Import from Traffic Capture

Simulations can be generated automatically from traffic captures obtained using the Crosscheck Proxy Server Traffic Capture tool. Captures generated from the proxy server tool will auto-create rules to emulate the traffic pattern based on matching requests. Timing can also be automatically replicated using the timing characteristics captured per each transaction request/response times.

When using this option, simulation rules will be automatically created to identify inbound requests and match them to the corresponding responses that were captured. Optionally, the user is prompted on import as to whether to also model the transaction timing characteristics, and if this option is chosen, the emulation will be build including delay tasks that use the values captured from the traffic capture.

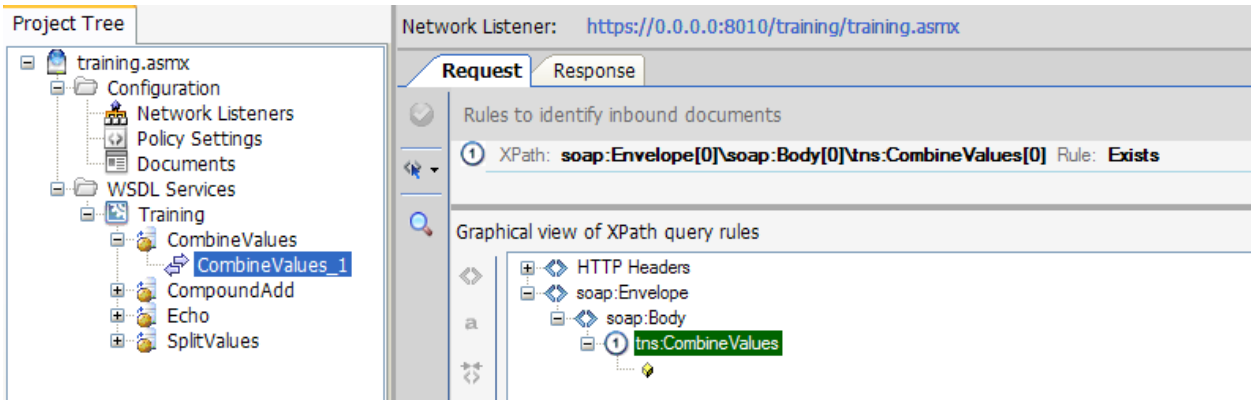


Simulation Request Processing

A simulator has two primary components. The request identification and processing, and the corresponding response document definition. The request processing consists of document identification rules, task processing, dynamic runtime variable capture, and functional success rules.

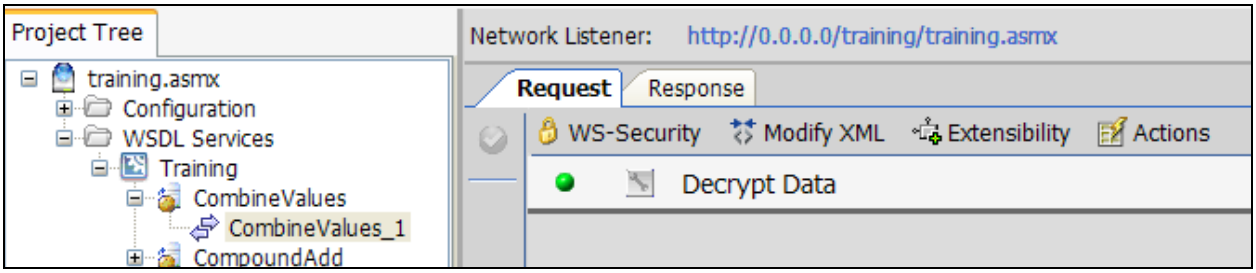
Simulation Request Processing – Document Identification

In order to trigger the simulator response, the inbound document needs to be uniquely identified by one or more Xpath rules that when a message is received to the simulation listener, it can be mapped to the current simulator for processing and response generation.



Simulation Request Tasks

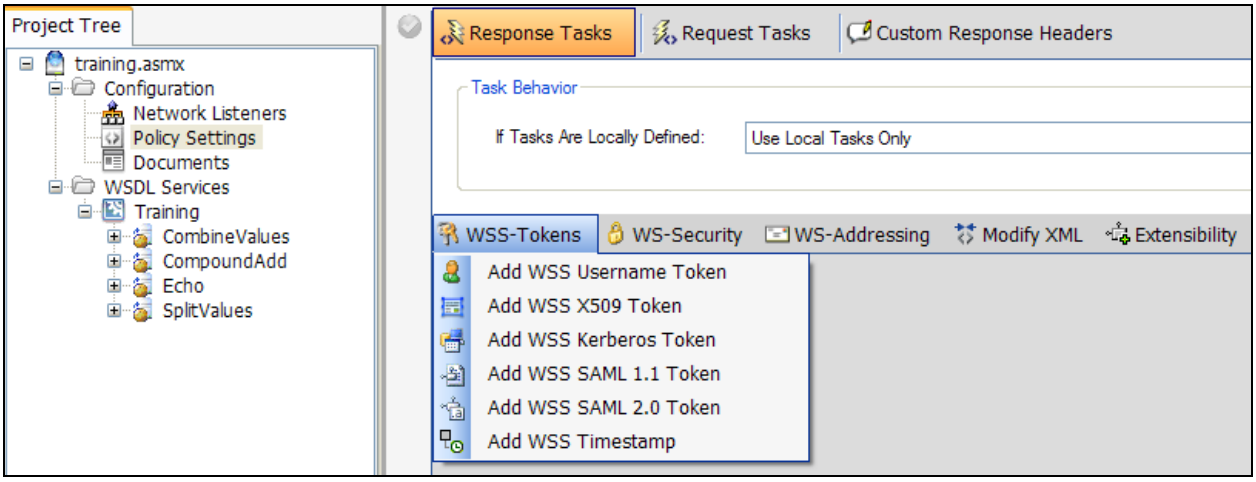
The tasks tab provides request document enrichment tasks that will be applied to the inbound request once it has been identified as a request targeted for this simulator. Tasks include security provions such as message decryption as well as custom rules, extensible API plugins, database manipulate, and other environmental or runtime actions.



Each time the Simulator is run, the corresponding task list is run against the inbound request.

Global Request Processing Tasks

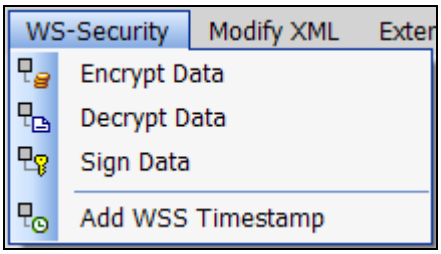
Common request processing tasks can be configured globally for each simulation project. The global task configuration is found on the Tasks and Settings tab when the top level node is selected.



When building lists of tasks, you can configure each one individually by clicking on the left icon and choosing **Configure**. You can view the resulting document with the result of invoking all tasks up to and including the currently selected one by clicking the left icon and selecting view. To change the order of tasks, click on the left icon and choose move up or move down. To remove a task, click on the left icon and choose Delete. If you hover over the labels for the tasks, you will see the settings summary for each displayed in the tooltip text.

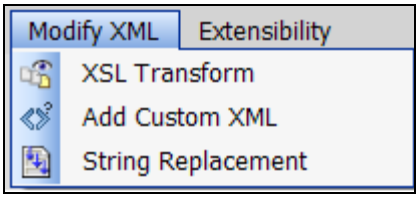
Request Task: Decrypt Data

The decrypt data task allows decrypting WS-Encrypted messages being returned from the back-end server. Configuration options include the key to use for decryption and whether or not to remove the header after decryption is complete.



Request Task Group: XML Transformation

The task options in this category include the full capabilities of XSLT transformation, direct string replacements, and adding custom XML fragments.



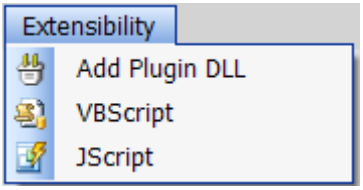
To add a task for manipulating the content or structure of the XML, click on the Modify XML menu and then choose the type of data processing to perform under the options listed. The selected task will initially be created un-configured, which is indicated by the red icon. Click on the left icon or on the task name itself to bring up the properties screen for the task. The properties screen includes configuration settings which are used by the task when it is dynamically applied to the request at runtime.

Once a task is configured, you can view the result of applying the task by clicking on the left icon and choosing the View menu item. This will invoke the all tasks in the current task list up to and including the selected one and show the resulting XML document with the tasks applied.

Request Task Group: Plug-in Extensibility

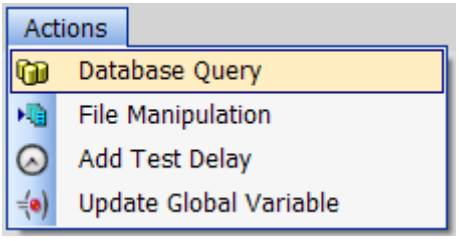
This task allows the modification to be performed via an external plug-in DLL or via inline scripting functions. You can write the plug-in in any language since CloudPort invokes the DLL using reflection to determine whether the function signature has been implemented..

Inline scripting is available in VBScript or JavaScript format. Inline scripting provides full ability to alter the request document and response headers prior to invoking the success criteria and runtime variable capture.



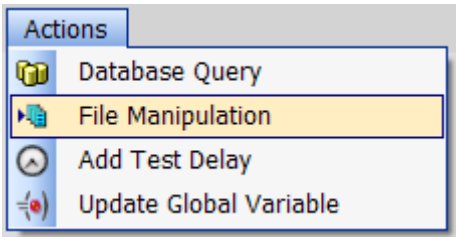
Request Task: Database Query

CloudPort provides the ability to run a database query using ODBC, or native SQL Server or Oracle database drivers. With this feature database tables can be created, deleted, or modified.



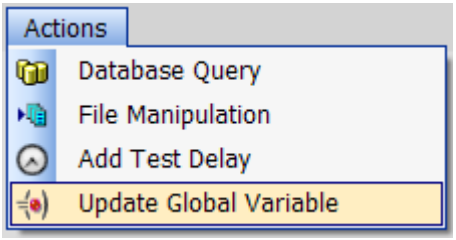
Request Task: File Manipulation

CloudPort provides the ability to create, append, or delete a file Variable substitution is enabled such that data to be written to a new file, or appended to a existing file can be static data, or dynamic data based on the current values for the runtime test.

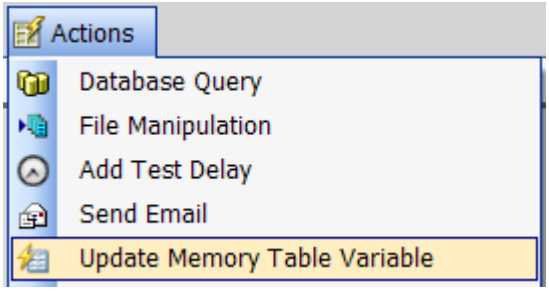


Request Task: Update Global Variable

Global variables can be updated while running a simulation to have future inbound requests running in the current runtime simulation use the update value(s) of the variables. This includes the ability to map global variable values to other test variables such as runtime variables, ADS variables, etc.



Request Task: Update Memory Table



Memory Tables provide the means to store complex data across test case calls and can be used anywhere that variables can be referenced to aggregate, preserve, or substitute information. In the test cases. The Update Memory Table task provides the means to do the following Memory Table operations:

SetValue

Set the Memory Table member variable to the specified value (which can also be a variable type). If this value was previously appended to (i.e. contains multiple entries), a SetValue call will clear the variable and replace the value with the specified value.

AppendValue

Append to the Memory Table member variable to the specified value (which can also be a variable type). Memory Table member variables can always be represented as multiple value entries (i.e. collections/arrays) simply by using the AppendValue method for the table.

GetValue
Gets the Memory Table member variable value. If this value is single value, then the call with return the single entry. If the variable contains multiple entries, then this call will result in a comma-delimited list of values. If the Memory Table variable was defined with an XML template, then this call will return with each value surrounded by the designated XML template

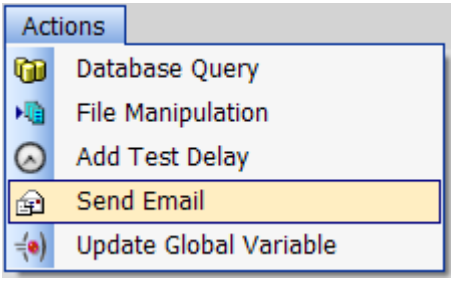
ReplaceValue
Replaces all instances of the find string with the replace string in the Memory Table variable.

ClearValue
Clears all values associated with a MemoryTable member variable

ClearTable
Clears all member variables in the specified Memory Table

Request Task: Send Email

To send an email when the simulator is invoked, you can create a Send Email task action from the Actions menu. Settings include the SMTP server with optional authentication, SSL, and the to, from, title, and body of the email. As is the case with all tasks, variables are enabled to parameterize the email contents with runtime variables.



Simulation Responses

Based on matching inbound identification rules, the response to the simulation can be defined. Responses include the ability to add security and identity provisions to the message as well as provide dynamic runtime variable extraction of inbound values or previous request values to include in the response. For HTTP and HTTPs simulations, the response is provided back on the network connection in which the client request was made. For MQ and JMS based simulations, the response is placed on the designated queue according to the message provider policy of the simulation.

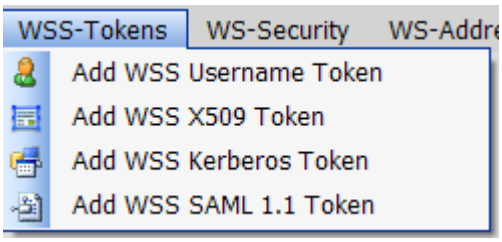
Response Tasks

If you are performing Success Criteria analysis of the response, or creating a Runtime Variable reference to response data, you may first need to manipulate the content of structure of the SOAP response dynamically before evaluating the Success Criteria or capturing the response value associated with the Runtime Variable. This can be accomplished through the definition of Response tasks.

Response tasks will be applied dynamically to the response before Success Criteria evaluation is performed and before the dynamic Runtime Variable data capture is done. To create response tasks, navigate to the Tasks tab in the Response area.

Response Task: SOAP Header Authentication (WS-Security Authentication Tokens)

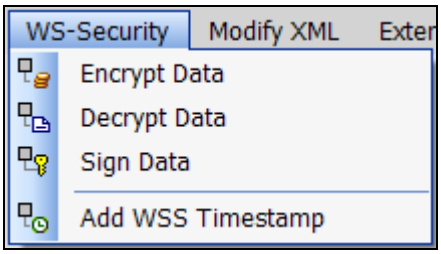
Adds a WS-Security 2004 SOAP header token according to the selected token profile specification. These tasks are used when the test environment supports message based authentication persistence and can be used to test and validate the Identity Management Authentication and Authorization decision.



To add a token identification task to the Simulator document, click on the WSS-Tokens menu and then choose the type of token task to create. Click on the task name to open the task properties screen for configuring the task. Once a task is configured, you can view the result of applying the task by clicking on the wrench icon and choosing the View option. This will result in invoking each task in the list up to the task associated with the wrench icon to preview the processing results that will occur each time the Simulator is executed. An XML viewer dialog is presented to review the resulting processed document.

Response Task: WS-Security Signature and WS-Security Encryption

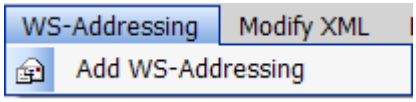
CloudPort supports the WS-Security Standards for generating compliant WSS-2004 SOAP Signatures and SOAP Encryption. These settings are required when testing in environments with data security requirements.



To add a Signature or Encryption task, click on the WS-Security menu. The Signature and Encryption tasks can be applied to the SOAP header, SOAP body, or any selected node. Click on the task name to open the task properties screen for configuring the task. Once a task is configured, you can view the result of applying the task by clicking on the wrench icon and choosing the View option. This will result in invoking each task in the list up to the task associated with the wrench icon to preview the processing results that will occur each time the Simulator is executed. An XML viewer dialog is presented to review the resulting processed document.

Response Task: WS-Addressing

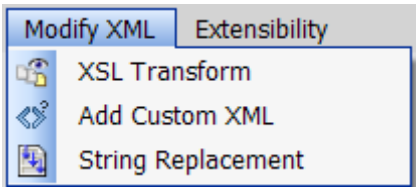
Adds a WS-Addressing header to the SOAP request.



To add a WS-Addressing task, click on the WS-Addressing menu and then choose the Add WS-Addressing option. Click on the task name to open the task properties screen for configuring the task. Once a task is configured, you can view the result of applying the task by clicking on the wrench icon and choosing the View option. This will result in invoking each task in the list up to the task associated with the wrench icon to preview the processing results that will occur each time the Simulator is executed. An XML viewer dialog is presented to review the resulting processed document.

Response Task: XML Transformation

Options to modify the request data via XSLT transformation, direct string replacement, and adding custom XML fragments to the request.

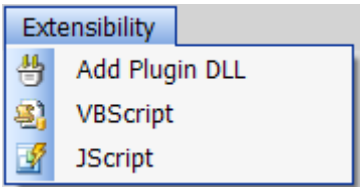


Click on the task name to open the task properties screen for configuring the task. Once a task is configured, you can view the result of applying the task by clicking on the wrench icon and choosing the View option. This will result in invoking each task in the list up to the task associated with the wrench icon to preview the processing results that will occur each time the Simulator is executed. An XML viewer dialog is presented to review the resulting processed document.

Response Task: Plug-in Extensibility

CloudPort supports custom DLL extensibility with a request event that can invoke DLL functions. This task option allows you to build your own DLL function to be invoked dynamically for each Simulator iteration being invoked. The DLL function can modify the request and HTTP header in any manner and the results of the modifications will be submitted as the Simulator request data.

The ICloudPortPlugin API interface for the DLL plug-in can be found in the plugins/ subdirectory under the installation directory. .

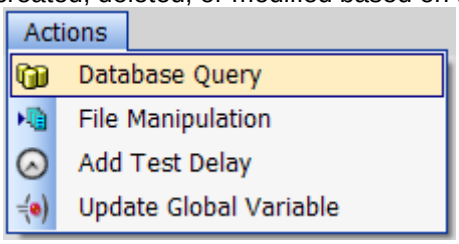


Inline scripting is available in VBScript or JavaScript format. Inline scripting provides full ability to alter the request and request headers within the scripting language of choice.

Click on the task name to open the task properties screen for configuring the task. Once a task is configured, you can view the result of applying the task by clicking on the wrench icon and choosing the View option. This will result in invoking each task in the list up to the task associated with the wrench icon to preview the processing results that will occur each time the Simulator is executed. An XML viewer dialog is presented to review the resulting processed document.

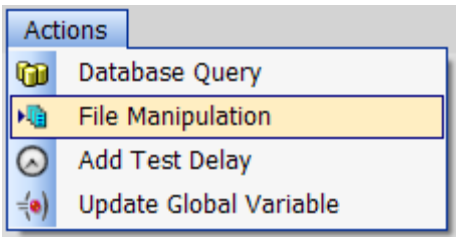
Response Task: Database Query

CloudPort provides the ability to run a database query using ODBC, or native SQL Server or Oracle database drivers. This action task fires prior to running the request. With this feature database tables can be created, deleted, or modified based on the SQL query before the test data is sent.



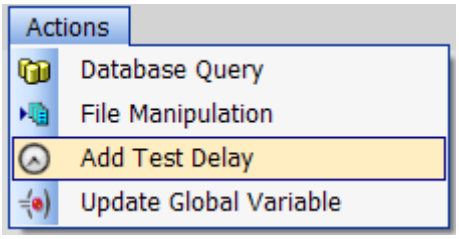
Response Task: File Manipulation

CloudPort provides the ability to create, append, or delete a file This action task fires prior to running the request. Variable substitution is enabled such that data to be written to a new file, or appended to a existing file can be static data, or dynamic data based on the current values for the runtime test.



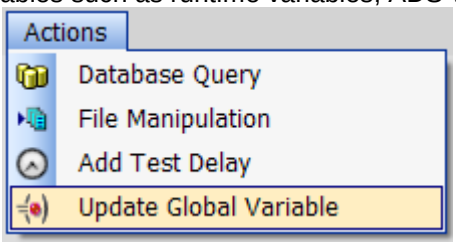
Response Task: Add Test Delay

To add a specific wait duration before the test is run, choose the Action->Add Test Delay and enter the delay value in milliseconds.

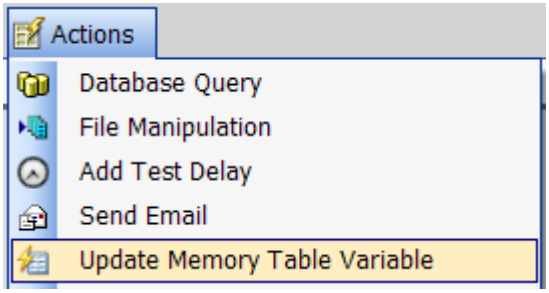


Response Task: Update Global Variable

Global variables can be updated while running a test to have future tests running in the current test suite use the update value(s) of the variables. This includes the ability to map global variable values to other test variables such as runtime variables, ADS variables, etc.



Response Task: Update Memory Table



Memory Tables provide the means to store complex data across test case calls and can be used anywhere that variables can be referenced to aggregate, preserve, or substitute information. In the test cases. The Update Memory Table task provides the means to do the following Memory Table operations:

SetValue

Set the Memory Table member variable to the specified value (which can also be a variable type). If this value was previously appended to (i.e. contains multiple entries), a SetValue call will clear the variable and replace the value with the specified value.

AppendValue
Append to the Memory Table member variable to the specified value (which can also be a variable type). Memory Table member variables can always be represented as multiple value entries (i.e. collections/arrays) simply by using the AppendValue method for the table.

GetValue
Gets the Memory Table member variable value. If this value is single value, then the call with return the single entry. If the variable contains multiple entries, then this call will result in a comma-delimited list of values. If the Memory Table variable was defined with an XML template, then this call will return with each value surrounded by the designated XML template

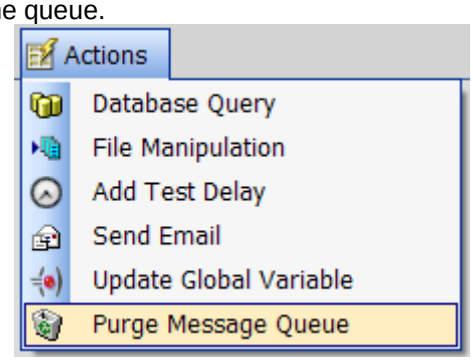
ReplaceValue
Replaces all instances of the find string with the replace string in the Memory Table variable.

ClearValue
Clears all values associated with a MemoryTable member variable

ClearTable
Clears all member variables in the specified Memory Table

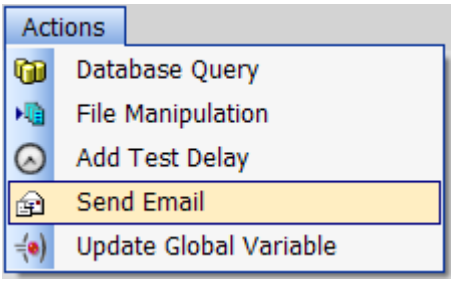
Response Task: Purge Message Queue

For JMS, Weblogic JMS, IBM MQ, or Tibco EMS Queues, the Purge Message Queue task can be used the clear the queue. This task will iterate through all messages on the target Queue and remove them from the queue.



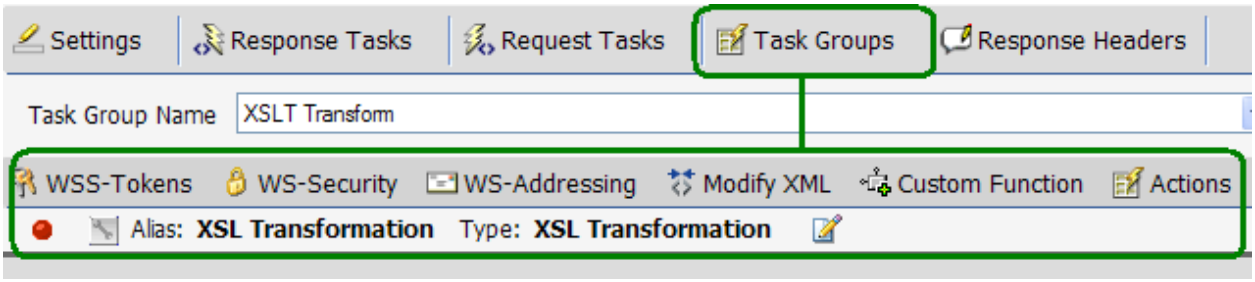
Response Task: Send Email

To send an email before a test request is sent, you can create a Send Email task action from the Actions menu. Settings include the SMTP server with optional authentication, SSL, and the to, from, title, and body of the email. As is the case with all tasks, variables are enabled to parameterize the email contents with runtime variables.



Global Task Groups

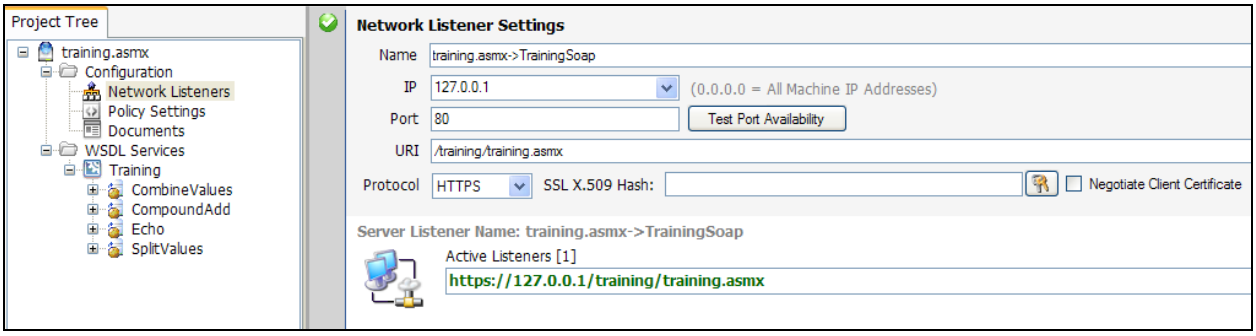
Tasks can be configured into re-usable task groups for association. Task groups are defined at the project level under the Policy Settings - > Task Groups section.



These tasks can be associated with response templates within the success criteria rule definitions to enable task processing of the template prior to sending the response back to the client.

Enabling SSL for Simulations

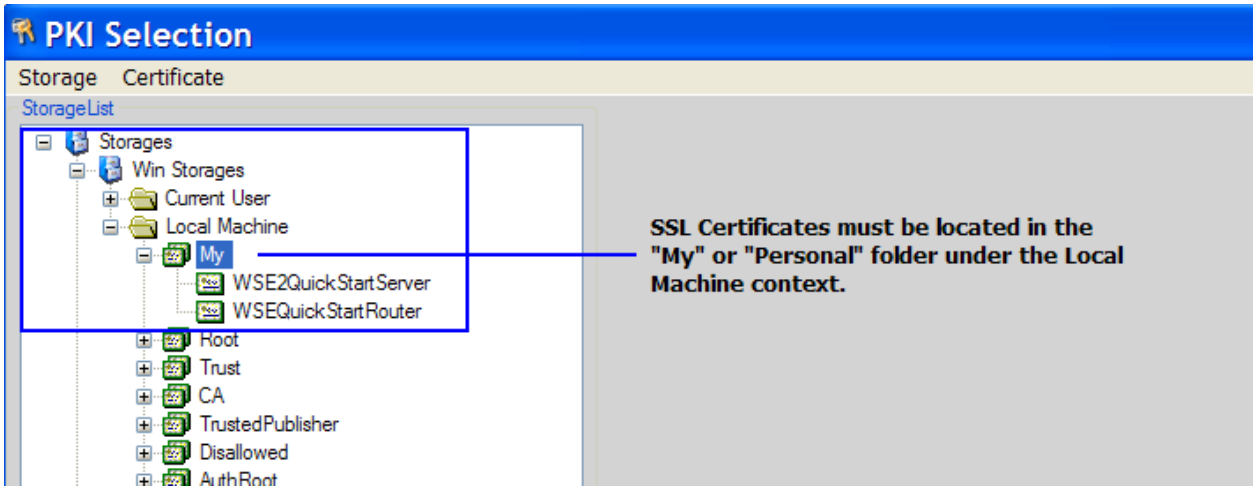
Endpoints can be enabled for HTTPS by selecting a certificate and choosing the option for whether to also require client X.509 authentication. The SSL listener certificates must be stored in the Personal folder on the Local Machine context in the Windows keystore. Click on the key icon to select the certificate to be used as the termination certificate.



Enabling Client Certificate 2-Way SSL

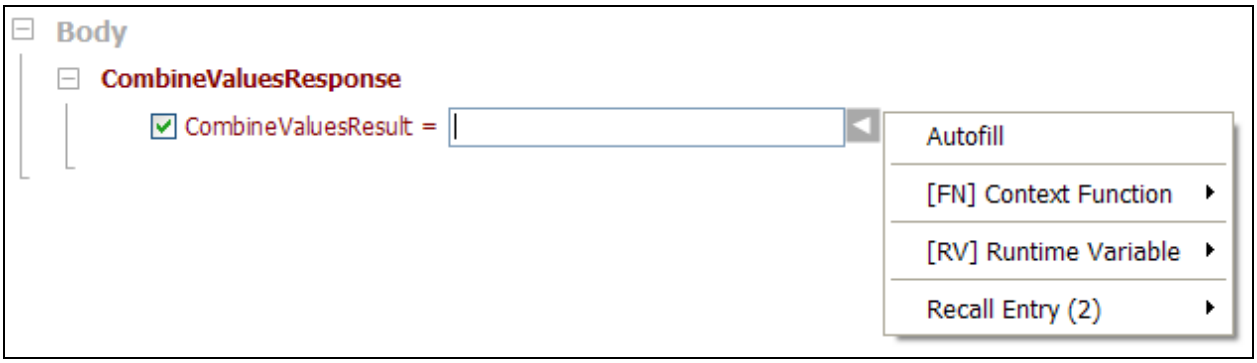
To enable 2-way SSL, click on the "**Negotiate Client Certificate**" checkbox which will then have CloudPort require a client X.509 certificate authentication for the SSL session.

SSL X.509 Hash – The hash identifier of the certificate to be used as the SSL termination certificate. This certificate must be stored in the Personal folder in the Local Machine context. This will appear as the "MY" folder under CloudPort.



USING VARIABLES

Simulators can be dynamically parameterized through the use of special variable references. Each time the test is run the variable is replaced with the current value(s) that the function variable type maps to.



Variables can be used in the request anywhere within the body content, the header, and task configuration fields. For dynamic response evaluation and synchronized variable replacement with inputs, variables can also be used to parameterize success criteria values.

Variable references appear as strings bounded by "$...$" with individual formats specific to each variable function type. The subtopic in this section define each type of variable functions, which include:

Context Function

Global Variables

Runtime Variable

Context Function Variables

Context Function variables are resolved for each request at the time it is being submitted to the back-end server. Context functions are useful for tasks such as tagging data with unique identifiers, populating date fields with the current date and time, generating unique IDs, and injecting static file contents into the XML document. Context function variables can be used within request data, headers, tasks, or success criteria.

Context Function Variable Format

Variable references for Context Functions are denoted using the following syntax:

$fn : Function (<optional parameters>) $

Where:

- **Function** is the type of function replacement to be performed
- **Optional Parameters** are parameters that may be sent to the context function (see examples below)

Context Functions

Now()

Each time the test is run the variable is replaced with the current date and time according to the optional format specifiers.

$fn:Now()$
Parameters:
(): Current date in ISO8601 format (`yyyy-MM-ddTHH:mm:ss.fffZ`)
(FormatString): Format for date. See Appendix A for supported data format values

RequestDocument()

This variable contains the complete request that was received. To create an echo service which simply returns the request back to the client, use this variable in the response.

$fn:request()$
Parameters: None

GUID()

Each time the test is run the variable is replaced with a globally unique identifier..

$fn:Guid()$
Parameters:
() – Standard length GUID
(Length) – GUID truncated to the value specified by the length parameter.

FileContents()

Each time the test is run the variable is replaced with the contents of the referenced file with optional xml encoding.

$fn:FileContents(<>,encode-xml-tags)$

Parameters:

(Filepath) – Full path to filename, encode any XML tags to > and <
(Filepath,true) – Full path to filename, encode any XML tags to > and <
(Filepath,false) – Full path to filename, no XML tag encoding

Random()

The Random() function will generate a new random value of type string, integer, or float. Parameters allow setting the data type, and the min and max value. For integer and float the min and max are used for the range of the generated random value. For string data type, the min and max parameters are used for the length of the string.

$fn:Random(TYPE,MIN,MAX)$

Parameters:

(TYPE) – String, Integer, or Float
(MIN) – Minimum value for the random values
(MAX) – Maximum value for the random values

B64()

Each time the test is run the variable is replaced with the contents of the referenced file with the contents base 64 encoded.

$fn:b64()$

Parameters:

(Filepath) – Full path to filename

MD5()

Each time the test is run the variable is replaced with the MD5 hash value applied to the contents of the string within the function parameter.

$fn:MD5(HashString)$

Parameters:

(HashString) – String to compute hash

SHA1()

Each time the test is run the variable is replaced with the SHA1 hash value applied to the contents of the string within the function parameter.

$fn:SHA1(HashString)$

Parameters:

(HashString) – String to compute hash

PEM()

When the function is initiated the variable is replaced with the PEM format value of the referenced X509 certificate selected from the native certificate selection dialog. Unlike other context functions, this is not a dynamic function, the PEM value is statically inserted when the function is selected.

$fn:PEM()$
Parameters: None (certificate browser will launch when this function is selected from the context menu.

Env()

This context function allows obtaining any Windows environment variable value to replace the variable name when the test is run

$fn:Env(<Environment Variable Name>)$
Parameters: Environment Variable Name

Global Variables

Context Function variables are resolved for each request at the time it is being submitted to the back-end server. Context functions are useful for tasks such as tagging data with unique identifiers, populating date fields with the current date and time, generating unique IDs, and injecting static file contents into the XML document. Context function variables can be used within request data, headers, tasks, or success criteria.

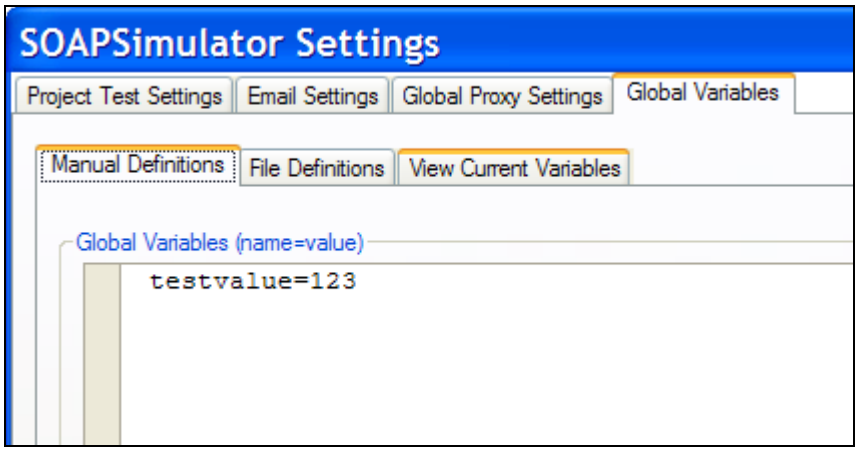
Global Variable Format

Variable references for Context Functions are denoted using the following syntax:

$fn : Global (<Global Variable Name>) $

Where:

- **Global** identified this as a global variable function
- **Global Variable Name** is the name of the Global Variable.



Global variables can be defined from the File->Settings dialog. Variable can be explicitly defined by name=value pairs, or you can point to one or more files which contain name=value pairs. The resulting compiled Global Variable file is stored in the <install dir>\lib\gvar.txt file. Thus, to alter the environment behavior of the CloudPort global variable settings, simply edit this file prior to running tests. Furthermore, this file can be shared with other CloudPort instances so that other instances can be synchronized with this instance's global variable settings.

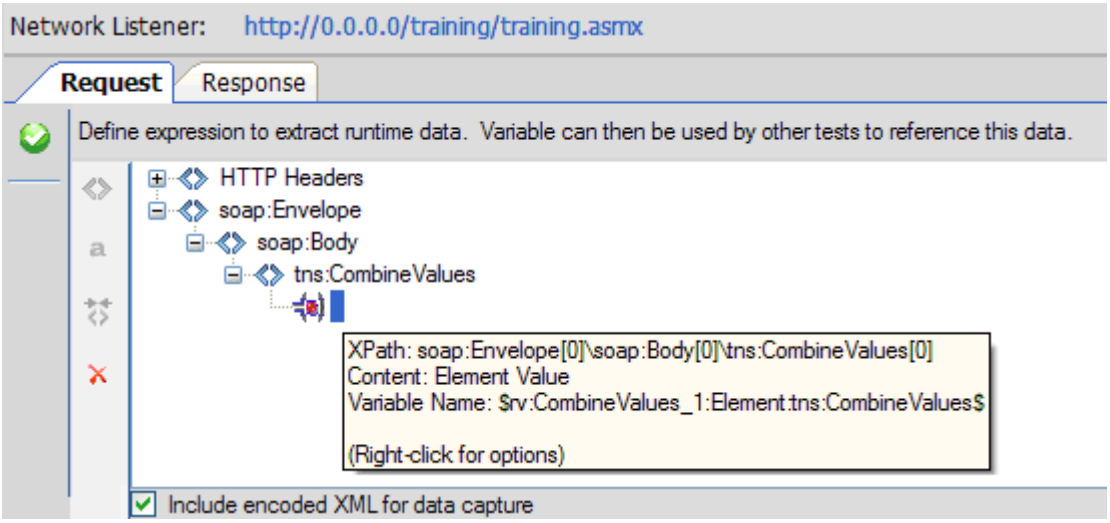
Runtime Variables

Runtime variables represent dynamic memory storage for Simulator request and response data when a test is executed at runtime. The resulting variable reference names can be used in other tests to create dependency sequences and allow values to persist across multiple tests.

Associating runtime variables from another Simulator creates a dependency chain. This provides the framework for Simulator chaining where results from one Simulator can be used as inputs to another Simulator. Runtime variables are used to dynamically capture request or response values at the time the test is run for value substitution in the referencing Simulator. Runtime variables can be associated across WSDL and XML project groups to chain multiple services or within the same project group to chain calls within the same end-point.

Runtime variables can be used in header, body, task field entries, or success criteria values in subsequent tests.

To create a Runtime Variable on a Simulator, go to the Runtime Variables tab on the Response panel and choose from the graphical view of the current response whether to create a reference to an XML fragment, and XML element value, or an XML attribute value. Name the Runtime Variable name.



Update Memory Table with Runtime Variable

Memory table variable offer flexible means of capturing Node, Element, or Attribute information and preserving it for later use. This variable option provides all of the existing capability of list variable and global variable mappings. The list and global variable settings remain for legacy configurations. See Memory Table variables for more information on how to use Memory Table variables to capture, preserve, aggregate, and apply functions to captured data.

By mapping the runtime dependency variable to a Memory Table variable, this provides the means to reference this value later in the test sequence, without creating a specific test dependency link to the current test. Not creating a specific test dependency link offers more flexibility to define the sequence of dependency, and the order of dependent test execution.

Store runtime values in List Variable

When enabled, this option will save each value running a running test suite in a list. This list can then be referenced from other test locations to retrieve values from the runtime variable list. The variable reference to extract the values back from a list is

$ rvlist:TestCaseName:CaptureValueType:VariableName(0)$

Where (0) is the index of the value to retrieve. Value indexes can be comma separated for multiple values such as

$ rvlist:TestCaseName:CaptureValueType:VariableName(0,1)$
$ rvlist:TestCaseName:CaptureValueType:VariableName(2,5,9)$

or the list variable can be used directly to collect all values stored in the list:

$ rvlist:TestCaseName:CaptureValueType:VariableName$

Mirror Value in Global Variable

When enabled, this option will store the runtime variable value in a global variable with the same name. This option is useful if you want to store a value and use it in another test case, but do not want an explicit dependency to be created on the runtime variable reference.

Include Encoded XML setting

When sending or receiving XML content in string encoded format, this setting will decode the XML within the existing document tags and allow variable capture within the encoded XML.

Runtime Variable Format

Runtime variables are resolved for each request at the time it is being submitted to the back-end server (i.e. at runtime). Runtime variables can be used within request data, headers, tasks, or success criteria.

Runtime Variables are denoted using the following syntax:

$ rv : TestCaseName : CaptureValueType : VariableName $

Where:

- **TestCaseName** is the name of the current Simulator which this variable is generated
- **CaptureValueType** is the type of data to be extracted from the defined location (value, attribute value, XML fragment)
- **VariableName** is the variable name used to help identify this variable for selection in other Simulators.

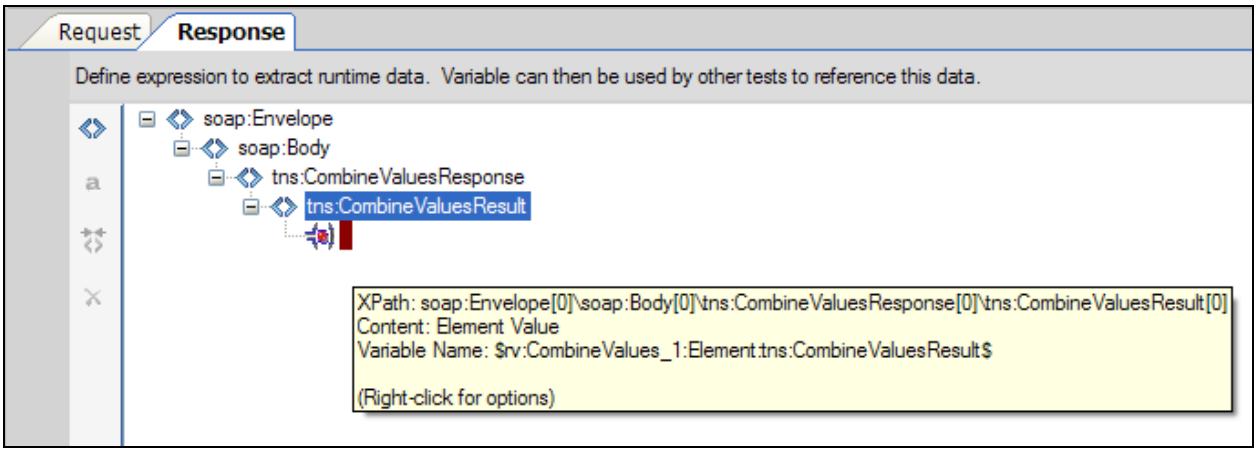
Runtime Variables can be configured to capture response HTTP header values, XML fragments, XML element values, and XML attribute values.

Capture / Replacement	HTTP Header Value	XML Attribute Value	XML Element Value	XML Node
HTTP Header Value	X	X	X	X
XML Attribute Value	X	X	X	X
XML Element Value	X	X	X	X
XML Node	X	X	X	X

The document structure and header structure are used to build the path designator to the location of the data that is to become the new parameter. The type of Runtime Variable can be any of the following:

Attribute: Runtime Variable path points to an XML attribute to extract the value from
Element: Runtime Variable path points to an XML element to extract the value from
Node: Runtime Variable path points to an XML node to extract the entire XML fragment from

The runtime variable definition area is found under the "Runtime Variables" tab in the Request and Response tabs. Selecting this tab will show the Runtime Variable capture screen.

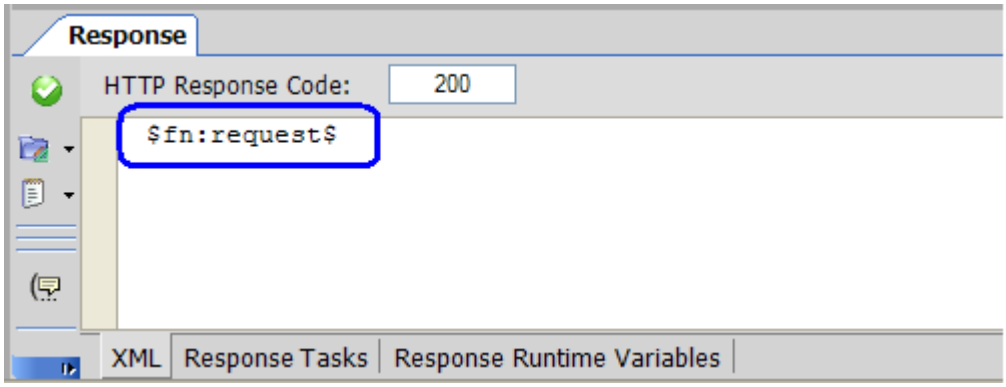


Once you have created at least 1 Runtime Variable in your project, it will be available for selection from any other Simulator in your project.

Special Variables

Echo Request Variable

A special variable is provided for creating reflection services that simply reflect the request back to the client. Each request is stored in the $fn:request$ variable and can be used to send the entire response back to client as follows:

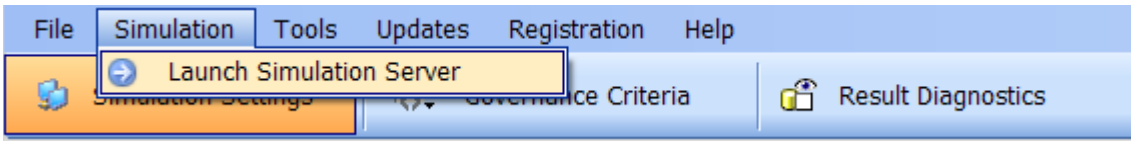

Response
HTTP Response Code:
200
$fn:request$
XML
Response Tasks
Response Runtime Variables

RUNNING SIMULATIONS

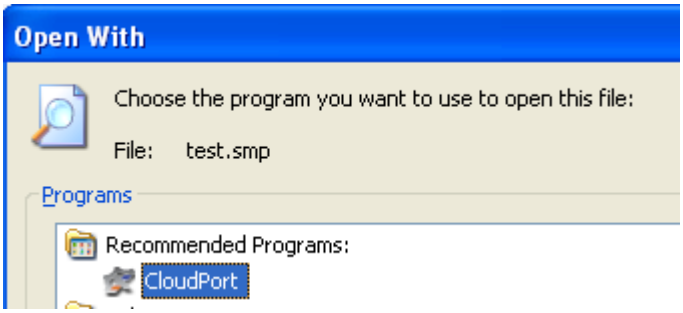
Simulations are run in a separate process and can be launched independently of the project console. A project file contains the settings for the simulation and the project file can be launched in one of the following ways.

Run using the local Simulation Server

Launch from the Simulation menu



Launch directly from the project file in Windows Explorer



Run CloudPortServer.exe and select a simulation rule file

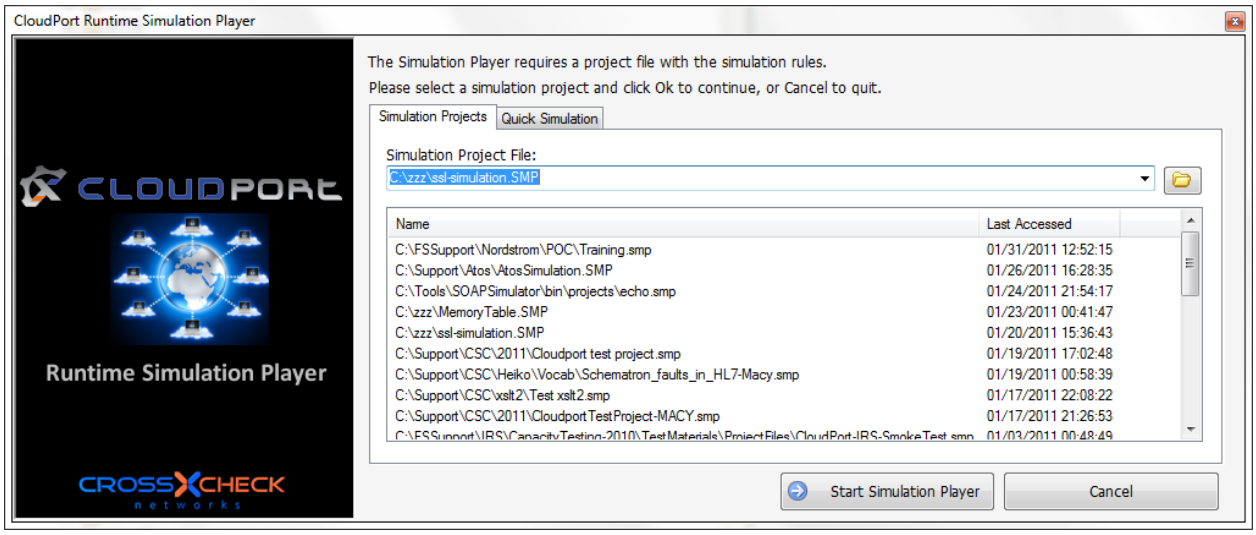
When launching the server directory, you will be prompted to load an existing project file

Simulation Projects

Simulation projects are created using the Simulation Editor provided with **CloudPort Platinum** and **CloudPort Standard** editions. Once the projects is published, it can then be run from any machine instance that has the runtime simulation player installed.

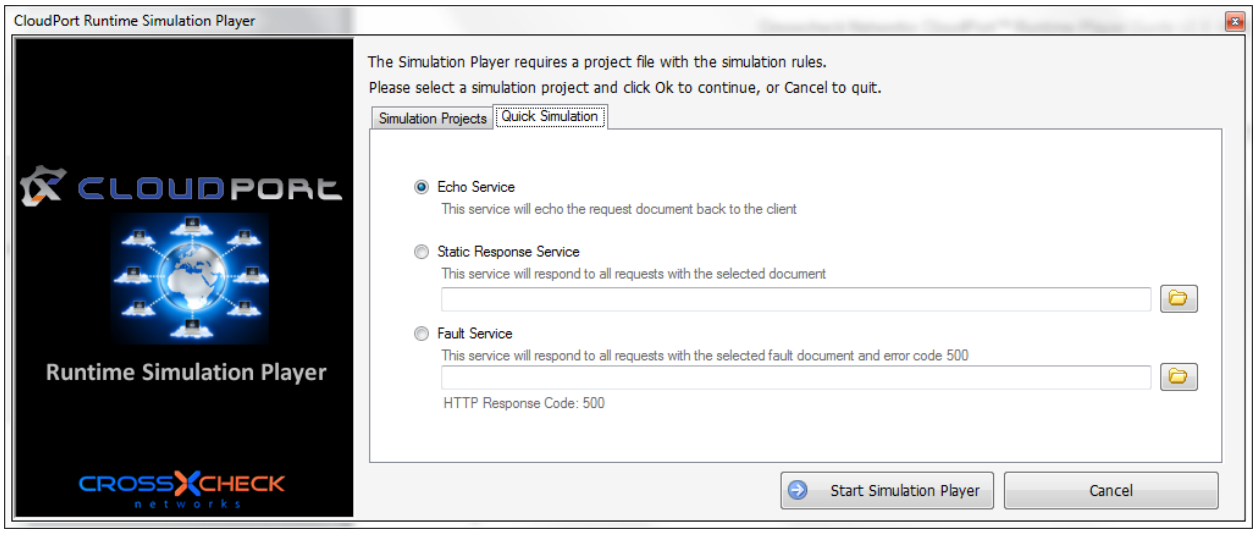
Loading a Simulation to Run

To run the simulation, simply select from a previously run simulation, or use the browse button to browse to the location where the **SMP** simulation project file is stored.



Embedded Simulations

There are several types of simulations that can be run without any configuration rules required.



Echo Service

This is an embedded service which will echo the request document back to the client. When running this simulation type you will be prompted for the port to run the echo service on.

Static Response Service

This is an embedded service which takes a specified document and always returns this document back to the client. When running this simulation type you will be prompted for the static response document and for the port to run the echo service on.

Fault Service

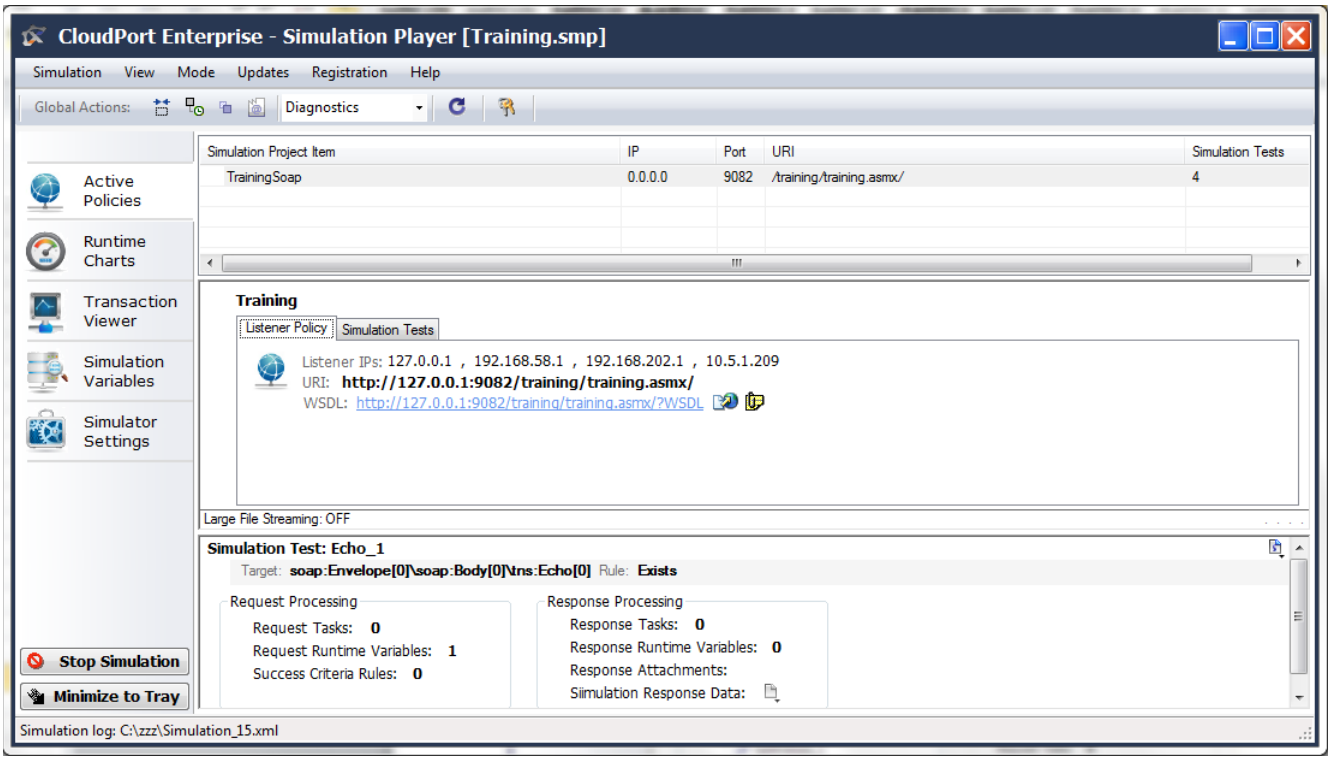
This is an embedded service which takes a specified document and always returns this document back to the client along with a 500 HTTP fault code. When running this simulation type you will be prompted for the static response document and the port the run the service on

Runtime Monitor – Active Policies

To see the current policies and network connections that are active for the simulation, click on the Active Policies panel. This will show the IP and ports that are active as well as the simulation rules that are enabled for traffic coming to these network listeners. If the simulation is WSDL based, this view will also show the dynamic links that are active for clients to retrieve the WSDL and Schema documents directly from the listener IP and port.

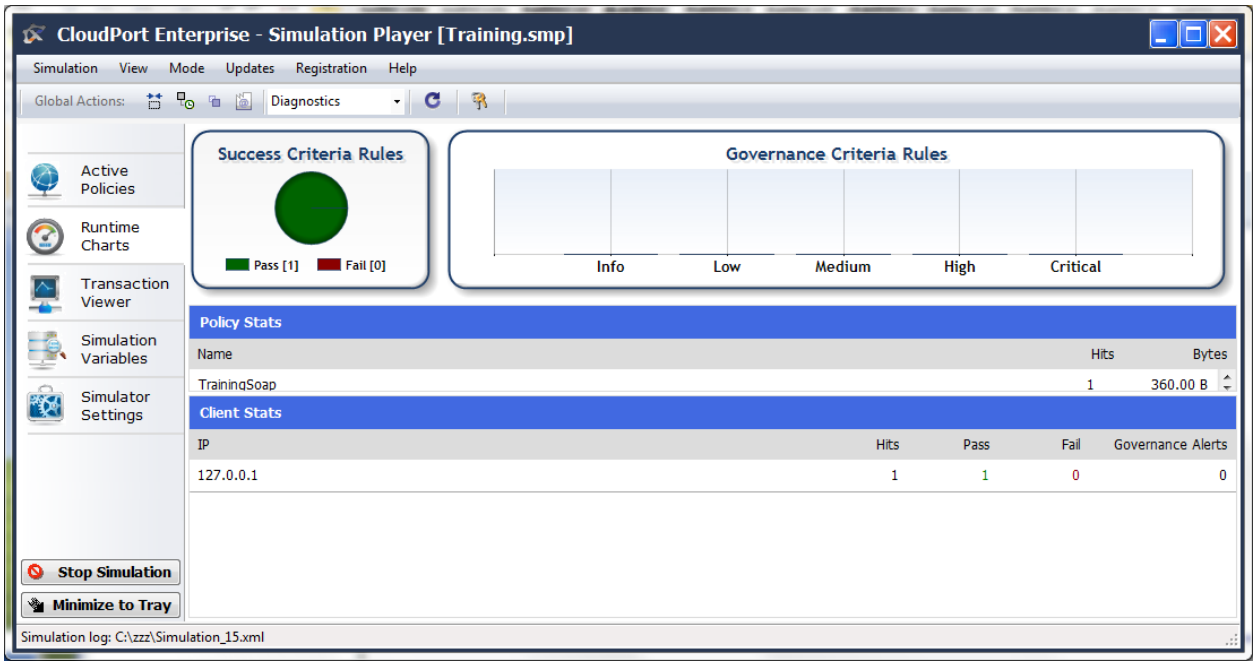
The simulation tests defined for the simulation will appear under the **Simulation Tests** tab in the middle panel. Click on a simulation project item to see the related network settings and tests. When clicking on a simulation test, the request processing and response processing settings are available for viewing in the bottom panel. If you are running using the free runtime player, the simulation rules can not be modified. However, if you are running using a floating license, you can modify the simulation rules dynamically at runtime. Note, however, that dynamic rule changes are only reflected in the running simulation. To change the underlying rules permanently, use the CloudPort Editor to save the changes into the CloudPort simulation project.

WSDL Simulation



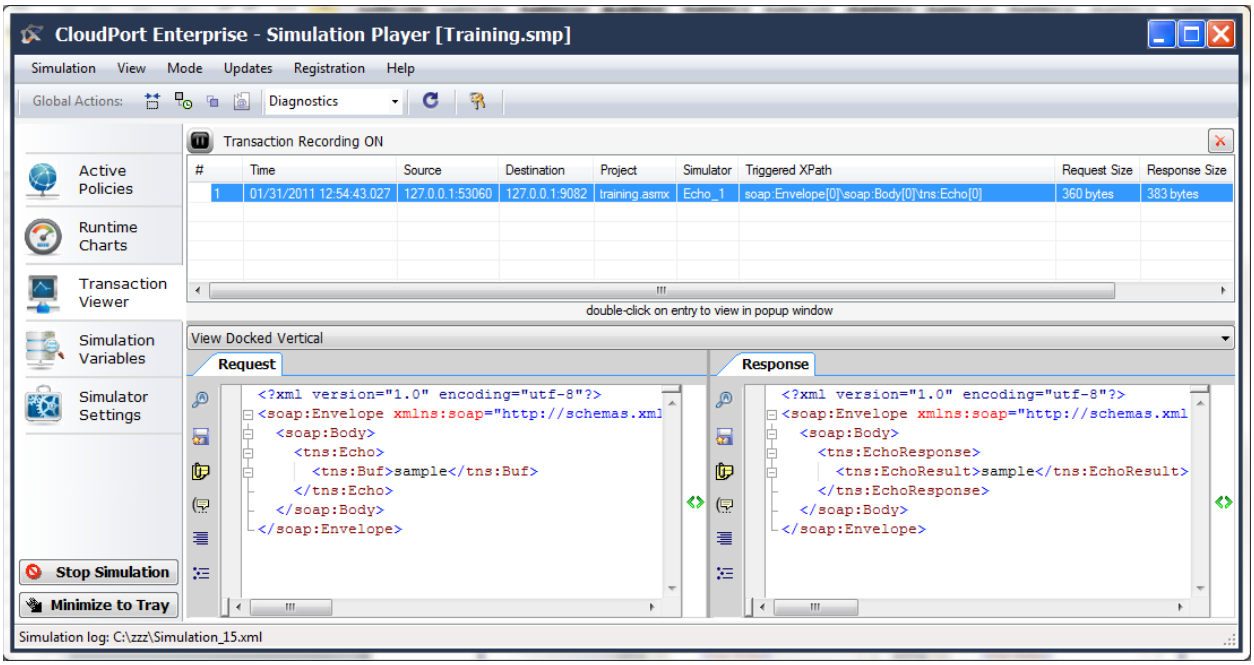
Runtime Monitor - Charts

The runtime charts show statistics for the current simulation session including functional success, governance alert triggers and transaction statistics per network listener and client IP.



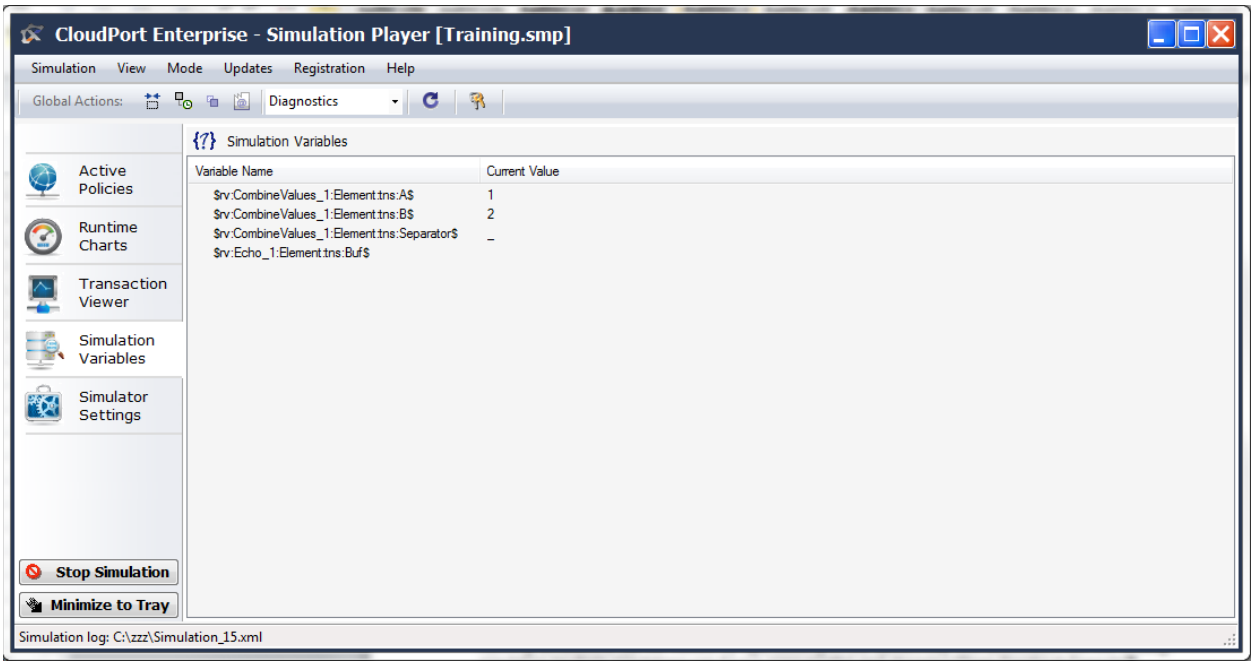
Runtime Monitor - Transaction Viewer

Transactions can be viewed in real-time from the Transaction Viewer panel. Click on an entry to see the request and response information for the transaction including request and response headers. Double-click on the entry to see the data in a separate popup window.



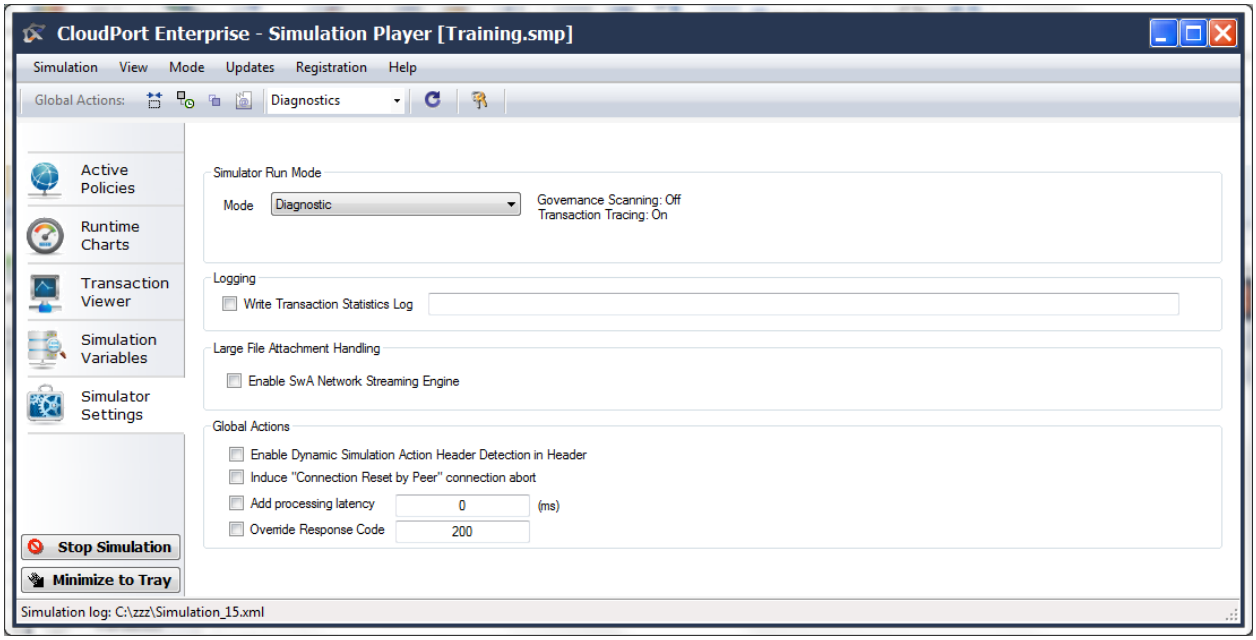
Runtime Monitor – Simulation Variables

If there are any variables enabled for the current running simulation, these will be shown on the Variables panel. The current values of the variables will be displayed and transactions that update the variable values will be shown on this panel.



Runtime Monitor – Simulator Player Settings

Settings for the current running simulation can be set from the **Simulator Settings** pane.



Simulator Run Modes

The run mode determines the devel of detail and information analysis that is to be performed on running transactions through the simulation player.

Governance Mode

This mode turns on transaction tracing to see all request and response information in the transaction viewer and turns on governance scanning of the inbound request. This mode is the slowest operating mode and is recommended only when performing governance scanning analysis of the inbound requests.

Diagnostic Mode

This mode turns on transaction tracing to see all request and response information in the transaction viewer and turns off governance scanning of the inbound request.

Performance Mode

This mode turns off transaction tracing and turns off governance scanning of the inbound request. This mode should be used when performing scalability testing and load testing of the client.

Transaction Statistics Logging

Separate logging to file is performed via the Write Transaction Statistics log checkbox.

Large File Streaming

The **SwA Network Streaming Engine** checkbox enables large file support for SwA MIME and MTOM by dynamically building the attachment messages and streaming the information from disk to socket. This dramatically reduces the memory overhead required to support scalability testing and concurrent client performance testing of SwA transactions.

Global Actions

Global actions affect all transactions across all loaded test cases. These settings can be used to force error conditions on the simulation player, or allow the Client Action Headers to be enabled such that the clients can have dynamic control over response timing and error states based on passing special Client Action Headers.

Enable Dynamic Simulation Action Header Detection in Header

This setting will enable the handling of Client Action Headers which are special processing headers that allows clients to control behavior such as time delays, error states, and simulated connection resets.

Induce Connection Reset by Peer connection abort

This setting will simulate connection aborts for all inbound connection while the setting is active. It can be used to simulate a server outage to assess how robustly the clients recover. By clearing the setting, transaction will once again be accepted. This setting can also be configured as a special variable allowing the client to control this setting per transaction using Client Action Headers.

Add Processing Latency

This setting will add additional latency to existing simulation rules that are triggered. This time will be added to any existing time already configured on the individual simulation rules. This setting can also be configured as a special variable allowing the client to control this setting per transaction using Client Action Headers.

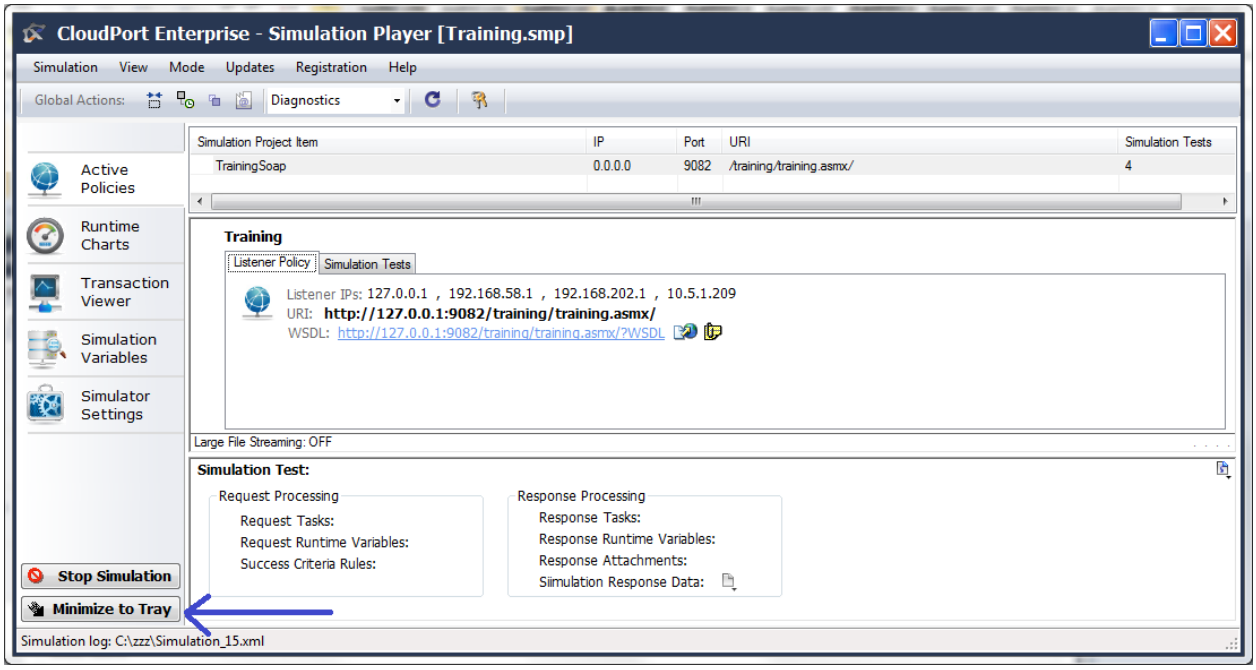
Override Response Code

This setting will force the response code to the configured setting. This can be used to alter response codes to determine how robustly a client handles various response codes. This setting can also be configured as a special variable allowing the client to control this setting per transaction using Client Action Headers.

Runtime Monitor – Opening and Closing

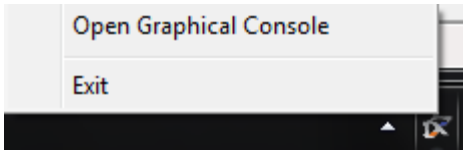
Runtime Monitor – Minimize to Tray

You can minimize the running simulation to the system tray by clicking on the Minimize to Tray icon. Running simulations can be viewed by right-clicking on the system tray icon associated with the running simulation (multiple simulations can be run concurrently) and choosing “Open Graphical Console”



Restore Simulation Monitor from Tray

To restore the graphical console from the running simulation in the system tray, right-click and select **Open Graphical Console.**



Stop a Running Simulation

To stop a running simulation, click on the **Stop Simulation** button from the runtime player console, or right-click on the tray icon in the system tray and choose **Exit**.

CLIENT ACTION HEADERS

CloudPort simulations allow the client to control certain aspects of the response behavior dynamically by sending the known action headers that the running simulation can detect and process. The supported action headers are detailed below.

Client Header: SIM-ResponseCode

SIM-ResponseCode
When this action header is detected in the inbound call, the provided response code will be used in the return call. Valid codes are in the range of 200-599.

Client Header: SIM-Delay

SIM-Delay / SIM-Wait
When this action header is detected in the inbound request, the simulator will wait the provided number of milliseconds before continuing with processing and response value.

Client Header: SIM-ConnectionReset

SIM-ConnectionReset
When this action header is detected in the inbound request, the simulator will abort the connection to simulate “broken pipe” and “connection reset by peer” network error scenarios.

SIMULATION TOOLS

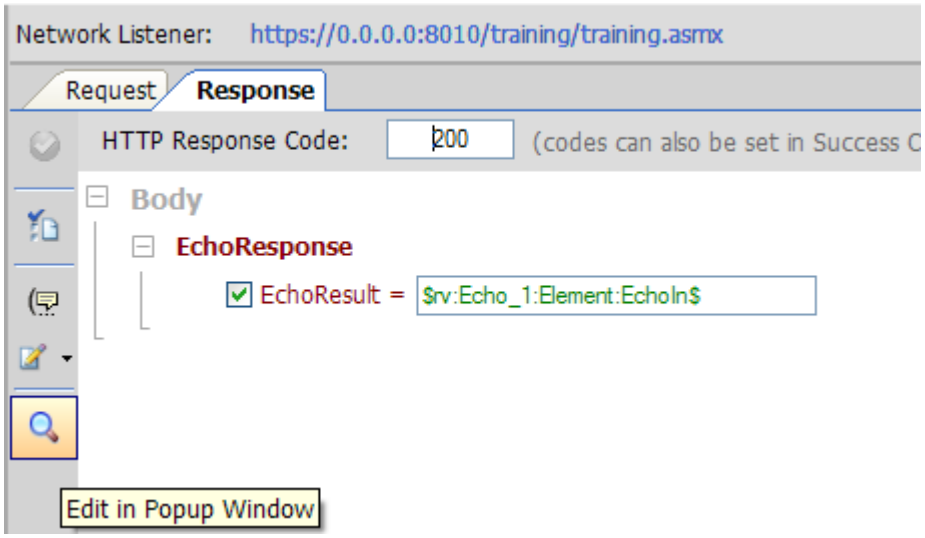
When authoring Simulators there are several configuration and viewing tools available to assist in the test generation process. Tools are available on the request and response toolbar as well as on the Tools menu.

Subtopics in this section include

- Edit Using Popup Window
- Run WSDL Schema Validator
- Native PKI Management
- UDDI Browser
- Auto Fill Schema Fields

Edit Using Popup Window

In Simulation Settings when editing a Simulator on the schema fields view or the XML view, the current edit screen can be viewed and edited in a separated window for more editing and viewing screen area.



Run WSDL Schema Validator

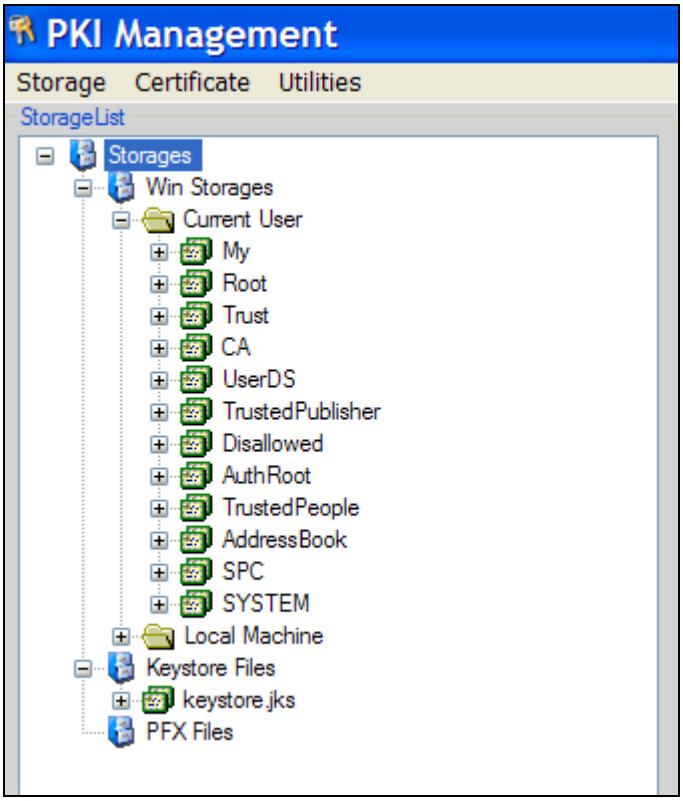
In Simulation Settings when editing Simulator requests or viewing test responses, this data can be validated against the WSDL schema to check that the structure and data integrity meet the requirements of the schema.

Native PKI Management

CloudPort provides integrated PKI management features to allow direct, native access to the windows keystore and also direct access to Java keystores. Anywhere you require access to a certificate or private key, an icon will be provided to access the native PKI interface to browse and select the keying material required.

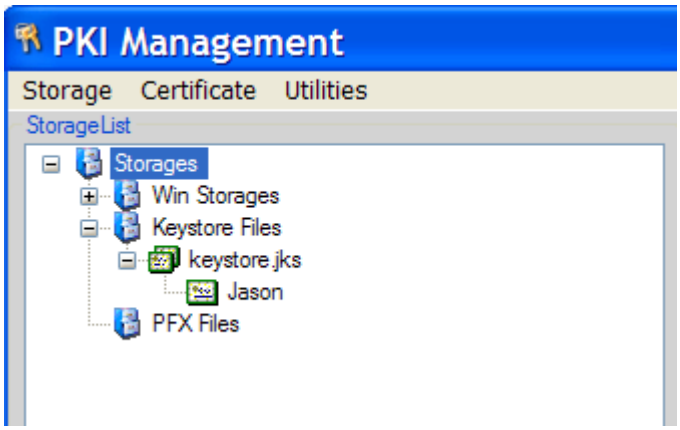
Windows Keystore

All keys in your windows keystore are presented for selection. You can also import, export, and generate keys in the "Win Storage" section.



Java Keystores

A Java keystore can be used to access public and private keys. To access keystore files, copy files with .jks extension to the CloudPort "keys" subdirectory. These will then appear under the "Keystore Files" folder when viewing the PKI screen. You will be prompted for a password to access the keys within the keystore.

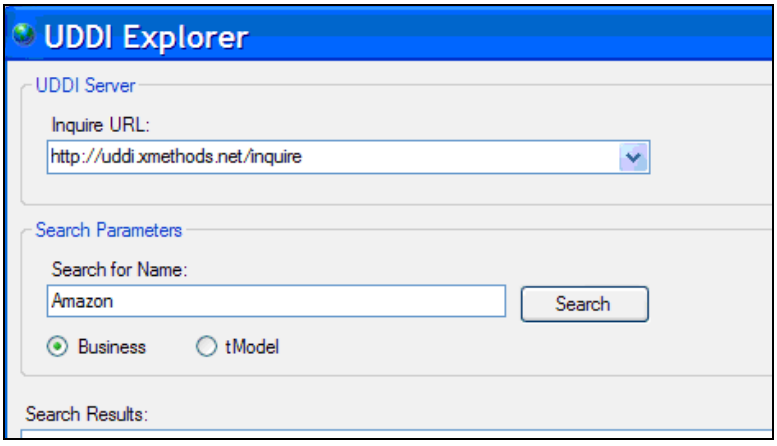


PFX Files

CloudPort provides access to file based PFX keys under the “PFX Files” folder. To access PFX files, copy files with .pfx extension to the CloudPort “keys” subdirectory. These will then appear under the “PFX Files” folder when viewing the PKI screen. You will be prompted for a password to access the private key of the PFX file.

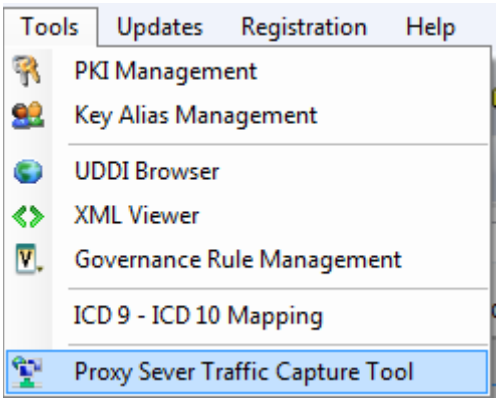
UDDI Explorer

For browsing a UDDI registry for services matching particular search criteria CloudPort provides a UDDI explorer from the Tools menu.



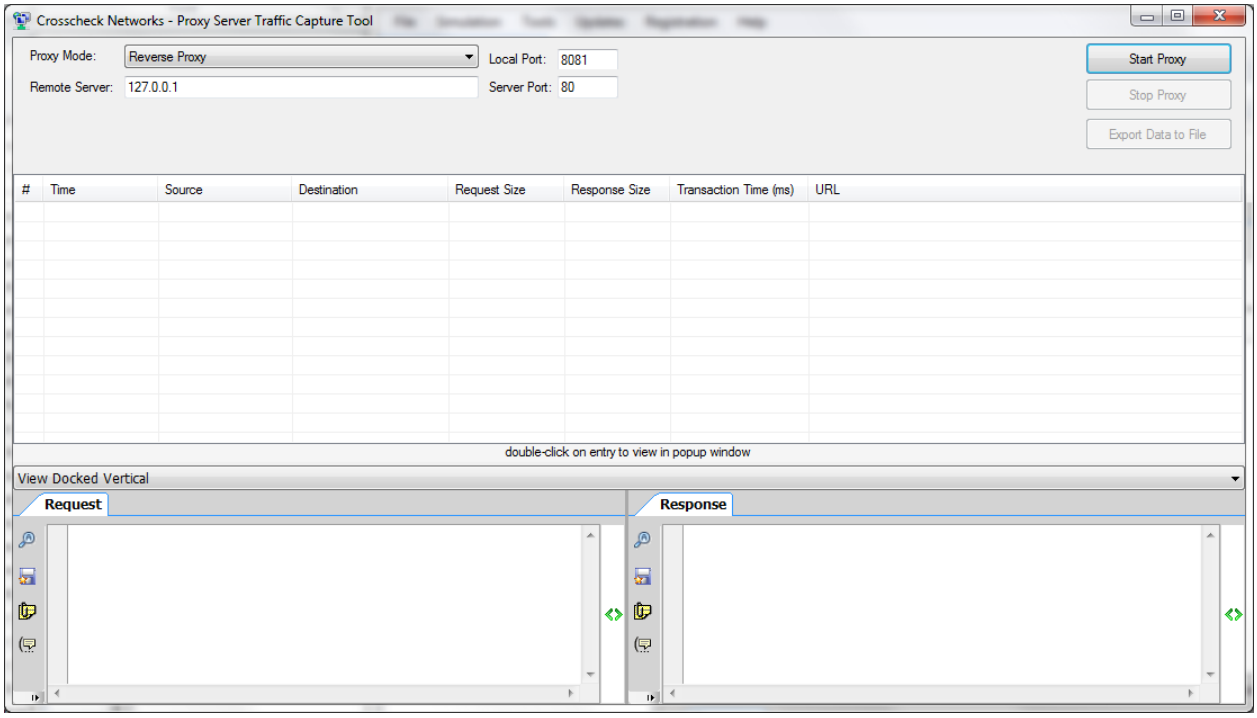
Proxy Server Traffic Capture Tool

To capture live traffic via forward HTTP proxy, or as a reverse proxy, you can start the Proxy Server Traffic Capture tool from the **Tools** menu.



The capture tool will listen on an IP address and a Port for inbound requests. The modes of operation are:

- Forward Proxy – Operates as a standard HTTP proxy
- Reverse Proxy – Becomes the endpoint to communicate with and forwards those requests to the actual server.



Once the capture is obtained, the results can be exported to an XML file which can then be imported into CloudPort via the Import->Proxy Server Traffic Capture

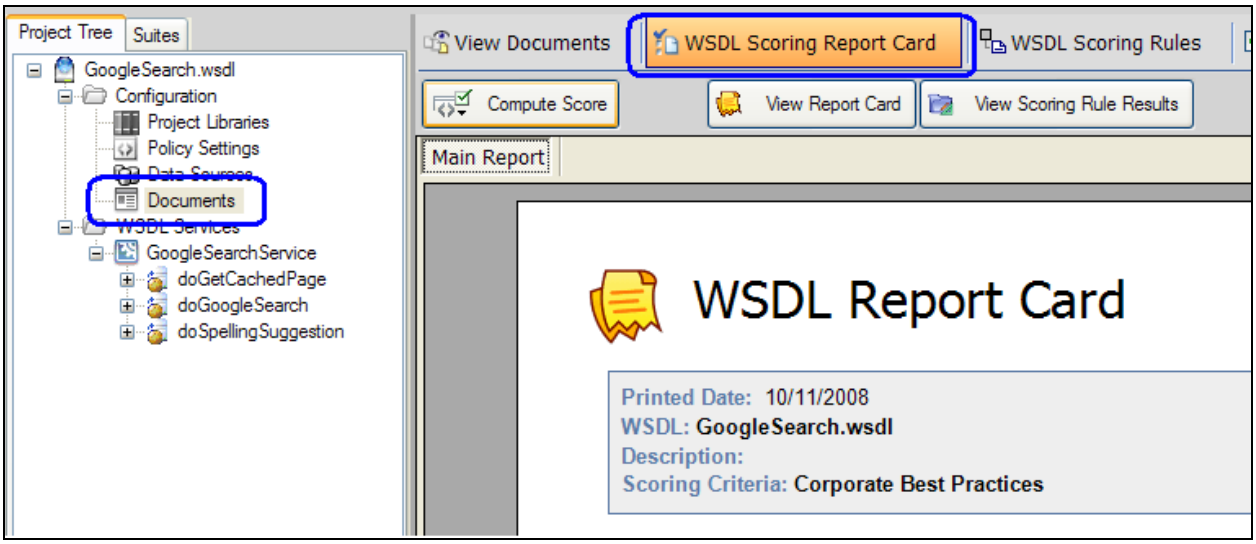
WSDL SCORING AND GOVERNANCE

CloudPort provides a framework for WSDL analysis and governance to help improve interoperability and best practices adherence in a consistent manner across the enterprise. WSDL Scoring allows WSDL and corresponding Schema to be scored across several categories to provide a scorecard of the rule results. The WSDL scoring rule set can be shared across other CloudPort instances to provide a corporate scoring framework to improve the quality of WSDLs and maximize interoperability.

Subtopics in this section include

WSDL Scoring

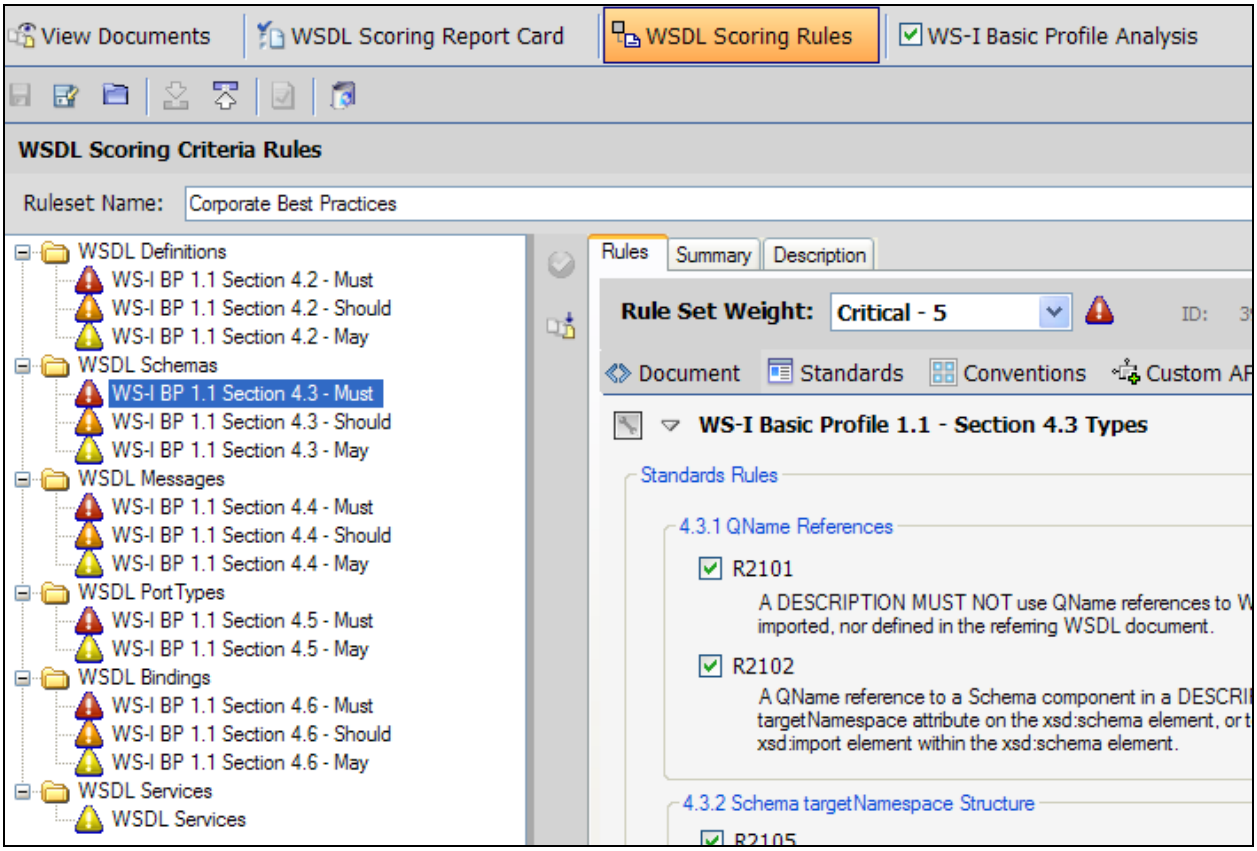
WSDL scoring evaluates the WSDL and Schema across the defined best-practices rules to build a scorecard report for the results. Once a WSDL is loaded into the project, the analysis options for the WSDL including generating a scorecard are found under the Documents node.



To compute the score of the WSDL, click on the “Compute Score” button. Once the scorecard is generated, the report can be viewed in a separate window by clicking on the “View Report Card” and “View Scoring Rule Results” buttons.

WSDL Scoring Rules

The rules for WSDL scoring are extensible and can be defined for the best practices based on industry standards, as well as specific to corporate governance and best practices. Rules are created with severity weights and each rule set has a summary and description which are used to describe the nature of the evaluation rules in the set. These descriptions appear in the scoring rule results report.



Rule Categories

Scoring rules can be defined for each component of the WSDL which are the WSDL Definitions, Schemas, Messages, PortTypes, Bindings, and Services sections of the WSDL. Within each of these categories, the rule sets include standards based rules such as the WS-I Basic Profile rule evaluation categories as well as extensible rules for naming conventions, and document based rules which provide full extensibility for isolating any part of the WSDL or schema. Rules also include extensible scripting of DLL plugin for even more customized analysis of each of the WSDL categories.

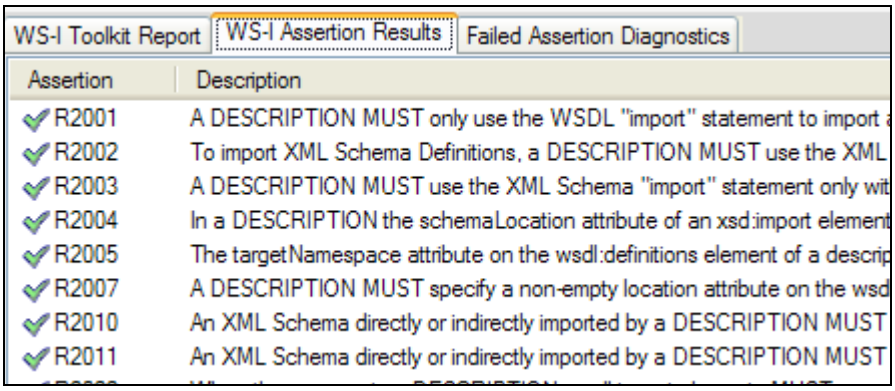
When configuring the rules that are based on WS-I category rules, each rule provides the ability to enable or disable the rules set forth by the WS-I Basic Profile 1.1. The set of enabled features on each rule will be evaluated and each will adhere to the severity weight specified. Severity weights vary from Info=1 through Critical=5.

WS-I Basic Profile Analysis

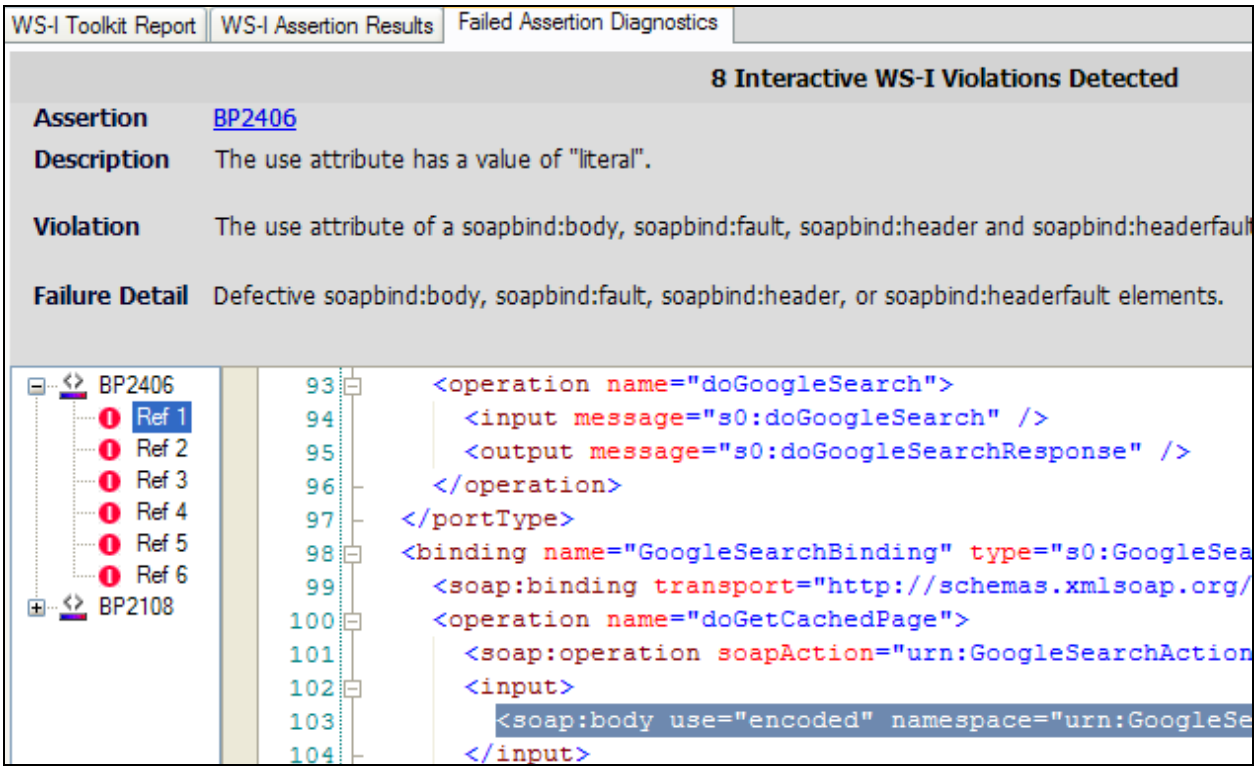
To perform WS-I Basic Profile scanning of the WSDL and Schema, go to the WS-I Basic Profile Analysis option on the menu under the Documents node.

To generate reports, click on the Analyze button after selecting the WS-I Basic profile to use for the analysis.

To look at the results of each assertion, click on the “WS-I Assertion Results” tab

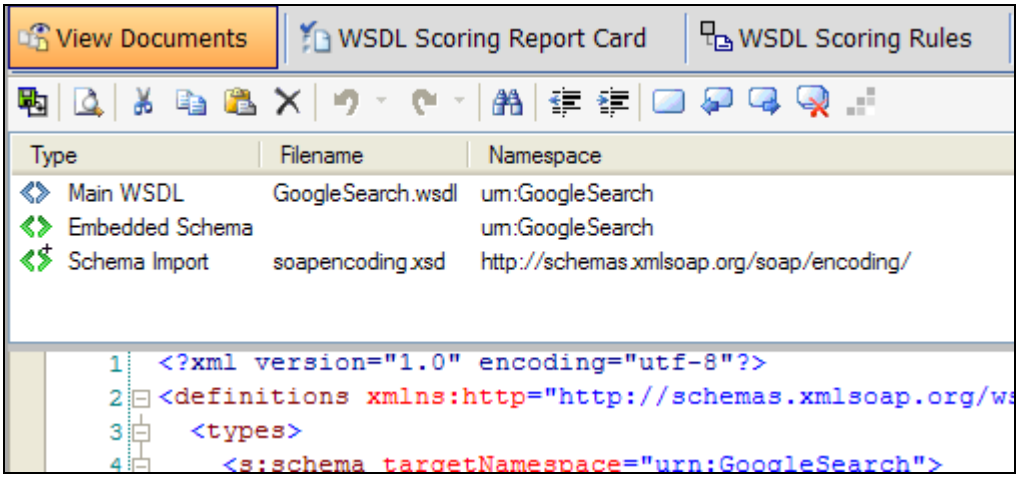


For failed assertions, if applicable based on the rule, you can click on the ‘Failed Assertion Diagnostics” tab to review the highlighted sections of the WSDL which triggered the assertion rule failures.



WSDL and Schema Document Review

Document review of any of the WSDL and Schema documents can be performed from the “View Documents” option menu under the Documents node.



PROJECT MANAGEMENT

All the information configured within Simulation Settings is portable and can be stored as a project file to be used to share among teams, version using source control, or used by the command-line interface to run simulations.

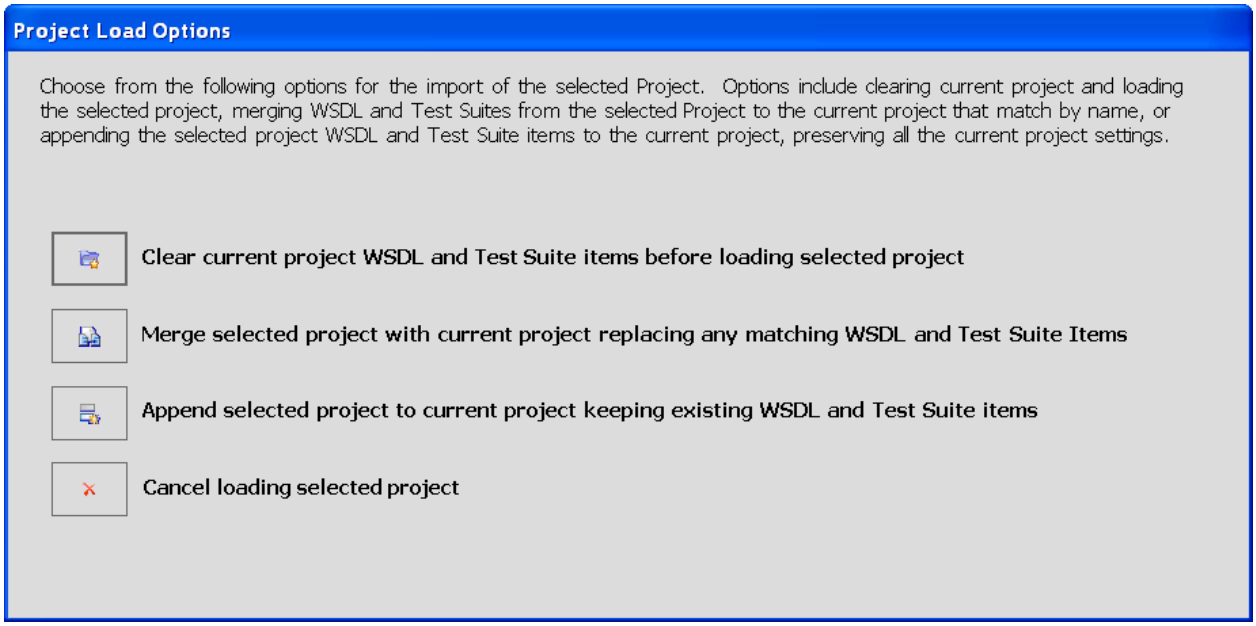
Subtopics in this section include

- Import Project
- Merge Project
- Append Project
- Save Project
- Upgrade WSDL in Project

Project management includes the features of loading, saving, exporting, and import project and WSDL files. Project files can be merged into the current project simply by loading an existing project while working on a project. You will be prompted to indicate whether you want to merge the incoming project with the existing projects, or clear existing settings before bringing in the selected project.

Import a Project

When loading a previously saved project, the option presented will depend on whether you currently have WSDL nodes defined. If you already have items loaded, you will be presented with several options on how you would like to load the project. These include:



Load as new Project

To load a new project, select the "**Create current project WSDL and Test Suite items"** option. This option will remove all settings and start with the loaded project only

Merge Project

To merge the selected project, select the "**Merge selected project with current project"** option. This option will search for all WSDL nodes that match those in the selected project and then merge all settings in the selected project with the settings currently loaded. This is useful when sharing projects and merging updates to the same project.

Append Project

To merge the selected project, select the "**Append selected project to current project"** option. This option will simply append all the selected project WSDL nodes to the end of the currently loaded ones. This setting is useful for combining projects into master projects.

Save a Project

At any time you can preserve the current settings into a project file. Options include the ability to save all tests and test suites, or selectively save a single WSDL project node and its related settings.

Save Full Project

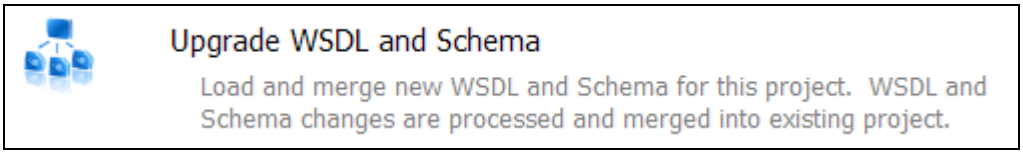
Choose File->Save Project to create a project file from the current settings.

Export WSDL Project

Individual WSDL projects can be exported by right-clicking on the WSDL node in the project tree and selecting the **Export to File** option.

Upgrade a WSDL

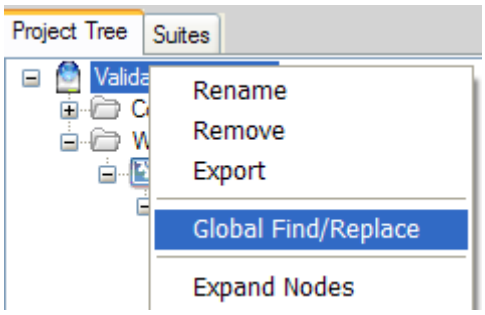
If you have a new version of a web service which has an updated WSDL document, CloudPort provides sophisticated WSDL merging capability which allows you to preserve existing project and Simulator settings when refreshing the WSDL in the project with later revision of the WSDL file. To update an existing WSDL in the project with a new revision, click on the "Merge WSDL" icon on the toolbar and then choose the location to obtain the new WSDL.

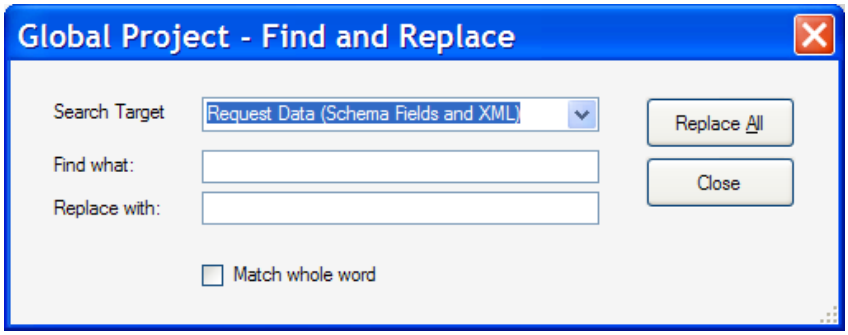


In the event that you load a WSDL that already exists in the project, you will automatically be prompted with options to load this WSDL as a separate and new instance to this project, or merge the WSDL with the existing WSDL. Once you choose to merge the WSDL, CloudPort will capture the schema differences between the 2 versions and automatically adjust your current Simulators with the applicable SOAP schema structure and type changes. Existing Simulator and WSDL enrichment settings will be preserved.

Global Find/Replace Options

You can right-click on the WSDL project node for the global find/replace option. This option will allow you to search and replace data across the entire WSDL project.





Choose the search target for the values of the project to search in and then provide the search criteria and replace criteria.

FUNCTIONAL SUCCESS RULES

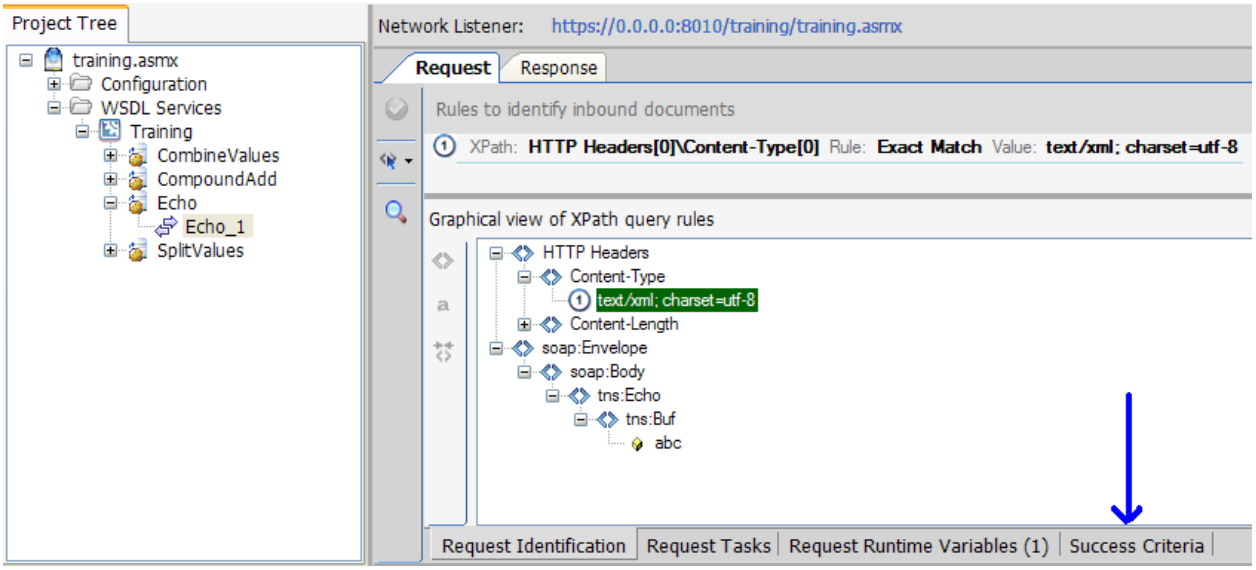
One of the key values of utilizing CloudPort is the ability to discover and resolve client integration and messaging issues before providing the client access to the actual production services. By using the Simulations and defining the inbound document success rules for each simulator, client implementation issues can be quickly identified and vetted.

Subtopics in this section include:

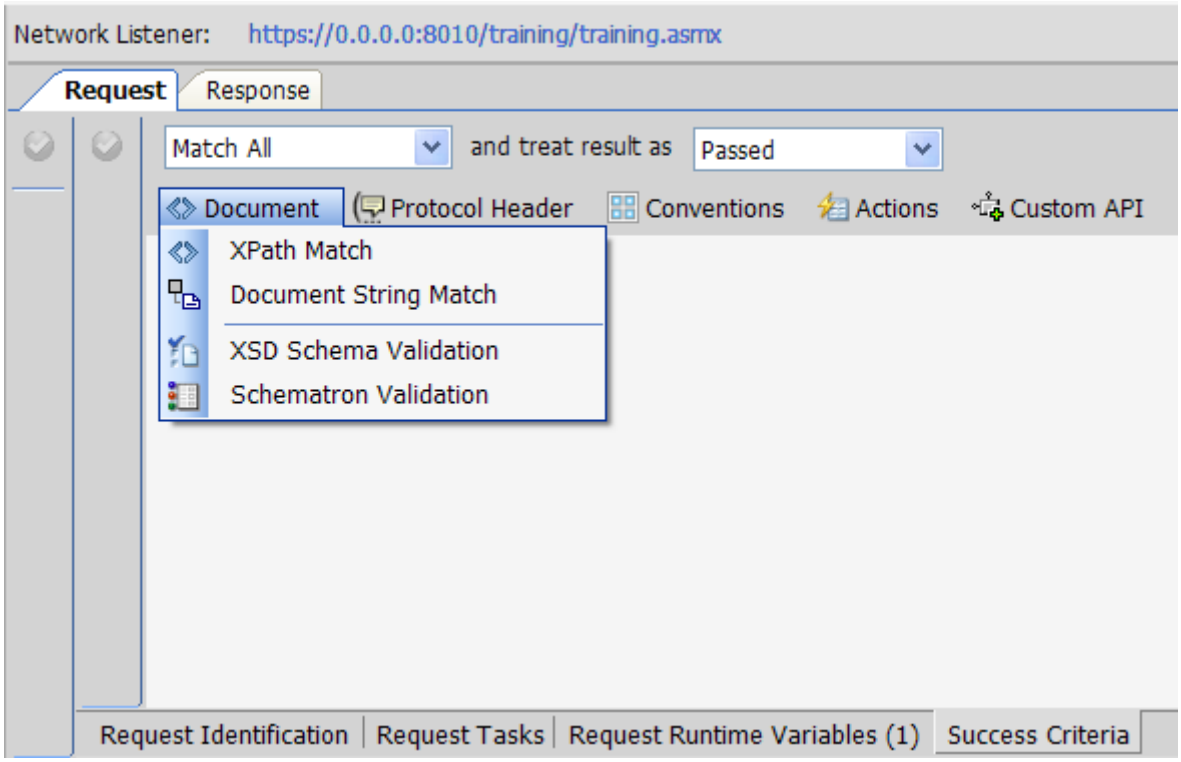
- Success Criteria Rules

Success Criteria Rules

The CloudPort Success Criteria evaluation engine provides a full range evaluation functions that allow you to define the analysis criteria for the inbound document to determine whether the request should be considered Pass or a Fail.



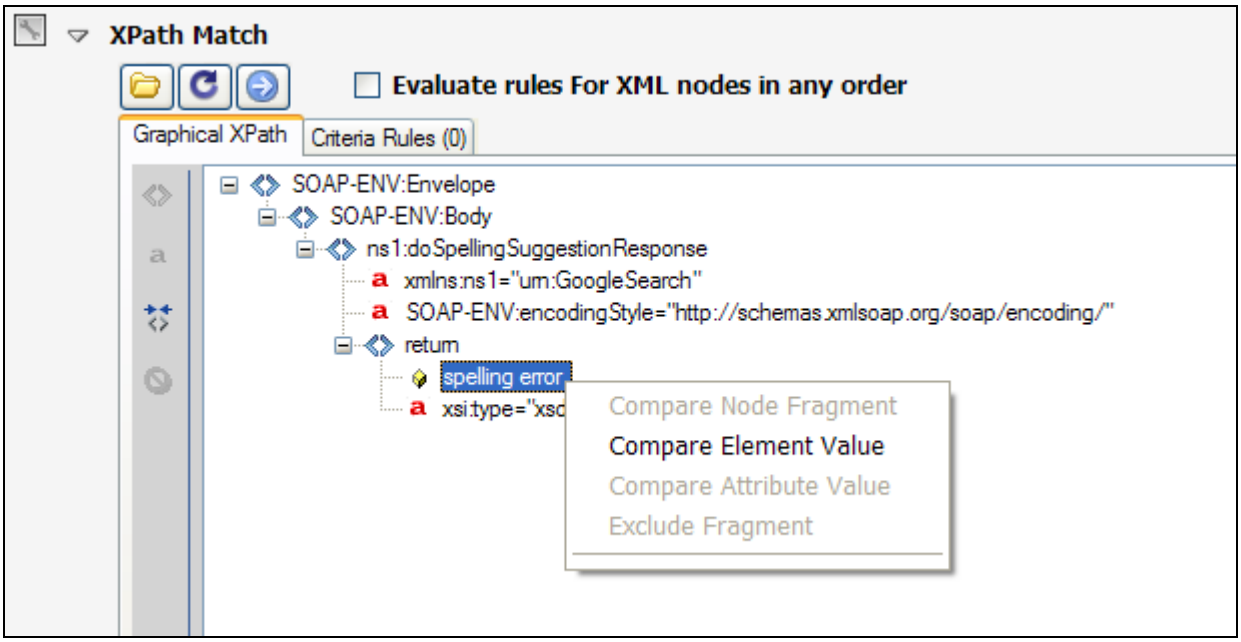
You can define as many criteria items as you want to accomplish a range of evaluation results. You can choose how you want to apply the set of criteria function in terms of matching all the items, matching any one of the defined items, or matching none of the items. The result of the matching criteria is then treated as configured to be a Pass or a Fail result.

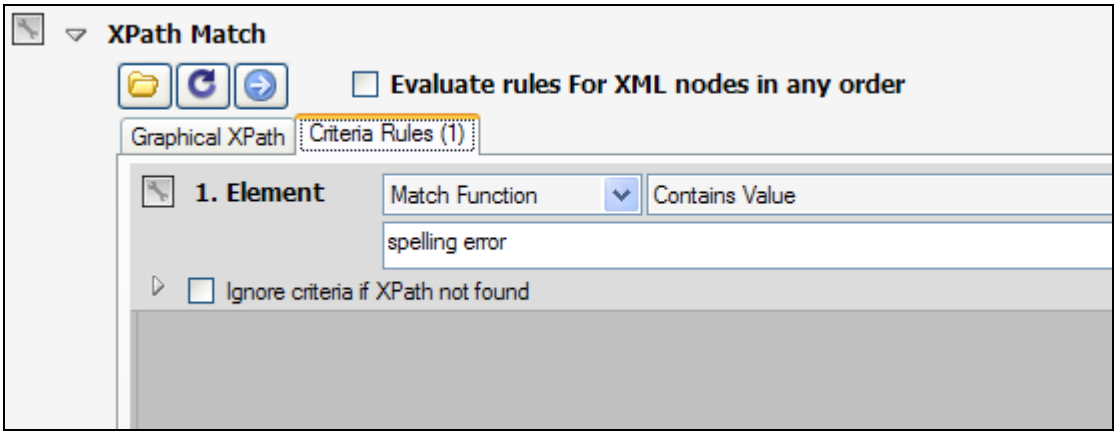


Value matching can be a substring match, a substring non-match, a regular expression pattern match, a numeric range, a string length range, or an enumeration of acceptable values. Additionally, each criteria specified can be set with an override action to force the result as a Pass or Fail.

Success Criteria Rules: XPath Match

The XPath Match rule provides the means to isolate parts of the data to compare against baseline stored results, specified values, or test variables (RV, ADS, etc). XML fragment Diff, XML element values, and XML attribute values can all be selected and rules created for comparison. Additionally, for array comparisons where the order of elements may differ from one test run to another, the XPath match criteria provides the option “Evaluate Rules for XML Nodes in Any Order”





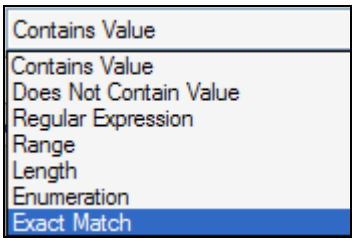
To configure the XPath Match Success Criteria, select Add->XPath Match. This will create a new Success Criteria rule with a Tree view representation of the current document. You can use the button on the top to invoke the web service to obtain the latest response document, or load a document from file. With the graphical representation of the request showing, you can choose one or more XML node, XML element value, or XML attribute value to create a rule comparison against. To select portions of the document to be evaluated, click on the XML node, element, or attribute value and then right-click, or select the active button on the left toolbar to create an XPath criteria rule for the selected XML.

Once an XPath criteria rule is created, the types of evaluation matching available are:

Match Baseline
Compare the live test request to the stored expected document at the XPath location(s) specified.

Match Function
Define the criteria to be used to evaluate the portions of the document selected. This criteria can be explicitly noted, or this criteria can be dynamically extracted from a database or Runtime Variable.

When match function is selected, the evaluation criteria options are:



Contains Value – This will search the referenced data for the substring value specified

Does Not Contain Value – This will search the referenced data for no instances of the substring value specified

Regular Expression – A RegEx pattern match criteria used to match the referenced data. Acceptable format for expression is any valid pattern match regular expression.

Range – A minimum numeric and maximum numeric value entered with a "|" delimiter as follows: MIN|MAX. For example, the value **23.4|23.4** would match the number 23.4, 23.40, and 23.400. The value **0|100** would match any number between 0 and 100.

Length – A minimum string length and a maximum string length entered with a "|" delimiter as follows:
MIN|MAX. For example, the value **5|5** would require a string of exact length 5. A value of **0|100** would match if the referenced data was between 0 and 100 characters.

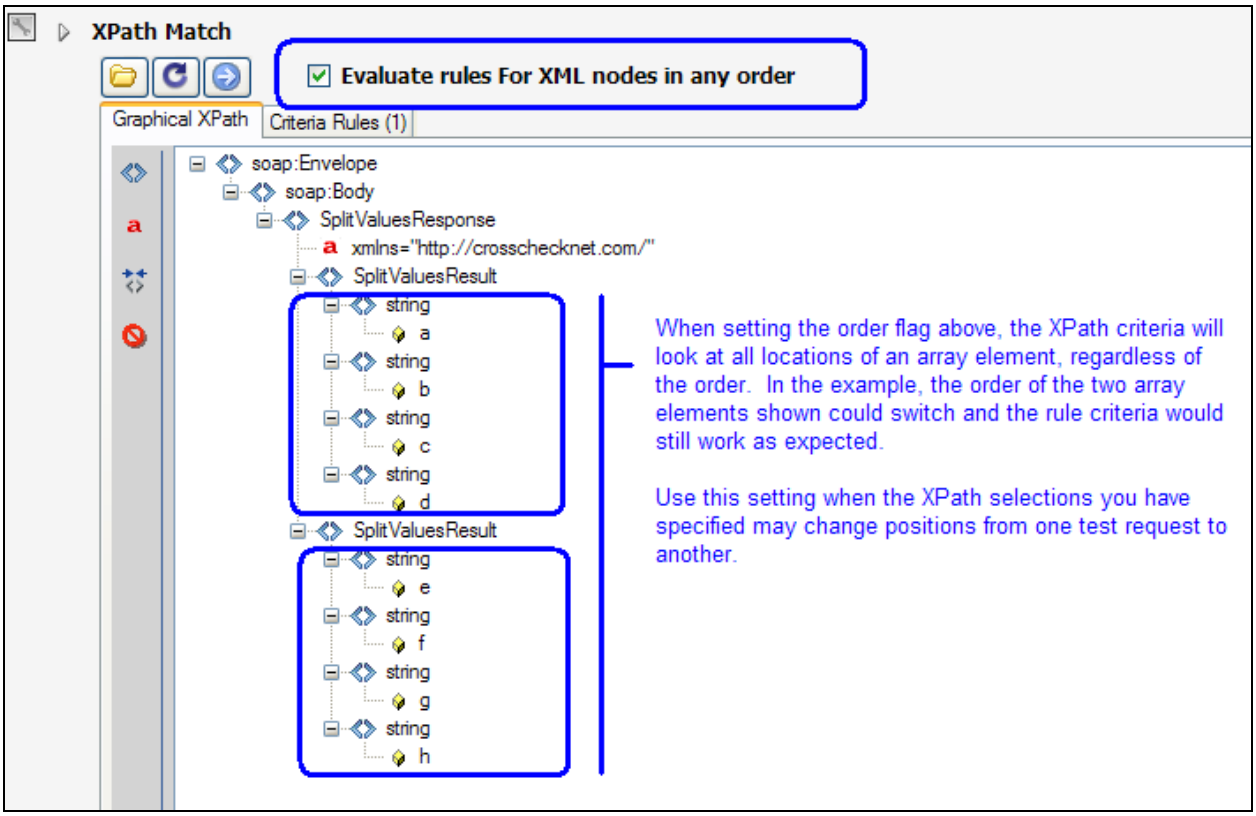
Enumeration – A set of acceptable data values entered with a "|" delimiter as follows: VALUE1|VALUE2|VALUE3|…|VALUEN. For example, the value **10.0|undef|none** would allow the referenced data value to be exactly "10.0", "undef", or "none".

Exact Match – This will compare the referenced data for the exact value specified

Tip: To maximize screen space when configuring rules, click on the rule title to open the rule configurable in a separate popup window.

Success Criteria Rules: XPath Match for XML Elements Changing Locations

The XPath match rules provides a setting to handle scenarios where the data specified changes locations within the XML document from one request to another. This can often be the case when dealing with collections or array elements where the value blocks appear in different locations. Setting the "Evaluate rules for XML nodes in any order" option will allow the XPath expression to search through all elements at that hierarchy level, regardless of order to evaluate the rule.



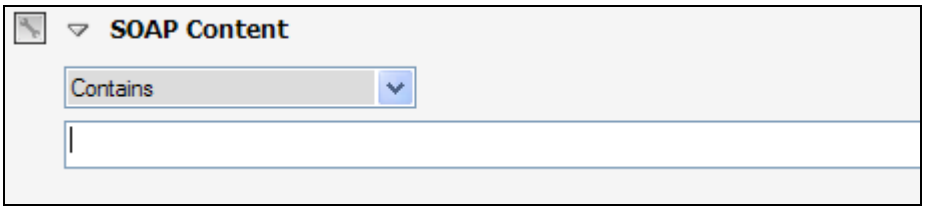
Success Criteria Rules: HTTP Header String Match

The HTTP Header criteria will evaluate the expression provided according to the match type specified. Contains will search for the presence of the string value in the HTTP header. Does Not Contain will ensure that the string pattern does not occur in the HTTP header, and Regular Expression allows you to specify a regex pattern match function to use to determine whether the regex pattern is found in the HTTP header. The value itself can be a static function, or it can be a dynamic value extracted from a database table or Runtime Variable.

Tip: To maximize screen space when configuring rules, click on the rule title to open the rule configurable in a separate popup window.

Success Criteria Rules: String Match

The Match String criteria will evaluate the expression provided according to the match type specified. Contains will search for the presence of the string value in the message data. Does Not Contain will ensure that the string pattern does not occur in the data, and Regular Expression allows you to specify a regex pattern match function to use to determine whether the regex pattern is found in the message. The value itself can be a static function, or it can be a dynamic value extracted from a database table or Runtime Variable.



Tip: To maximize screen space when configuring rules, click on the rule title to open the rule configurable in a separate popup window.

Success Criteria Rules: VBScript Module

The VBScript Module allows you to use inline VBScript functions to evaluate the data.

You can define the script directly using the VBScript editor and you can also leverage existing functions and libraries by including these files using the Manage Includes dialog. A sample code snippet is included in the editor when you click on the Edit Script Code button.

The VBScript interface calls the defined function “Sub Evaluate(unused)”. Variables are passed to the VBScript environment through a global variable called Parameters which includes the following member variables:

Parameters.VariableNames: array containing names of any existing active variables
Parameters.VariableValues: array containing values of any existing active variables
Parameters.Request: The current request
Parameters.RequestHeader: the current request header
Parameters.Response: The current response
Parameters.ResponseHeader: The current response header

Success Criteria Rules: JScript Module

The JScript Module allows you to use inline JScript functions to evaluate the response data.

You can define the script directly using the JScript editor and you can also leverage existing functions and libraries by including these files using the Manage Includes dialog. A sample code snippet is included in the editor when you click on the Edit Script Code button.

The JScript interface calls the defined function “void Evaluate”. Variables are passed to the JScript environment through a global variable called Parameters which includes the following member variables:

Parameters.VariableNames: array containing names of any existing active variables
Parameters.VariableValues: array containing values of any existing active variables
Parameters.Request: The current request
Parameters.RequestHeader: the current request header
Parameters.Response: The current response
Parameters.ResponseHeader: The current response header

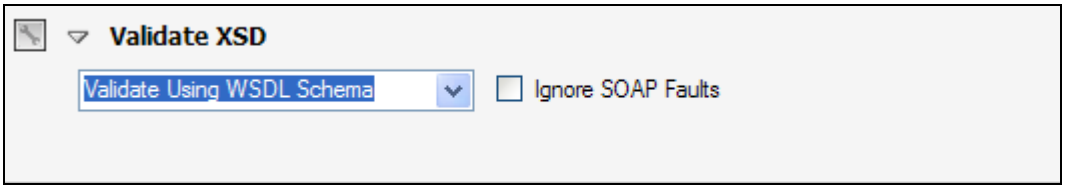
Additionally, Memory Table features are integrated directly within the JScript modules. You can create, modify, set, append, and clear Memory Table variables.

A Memory Table variable action command can be created from the Memory Table menu, or whenever you type “MemoryTable.” from the script editor.

MemoryTable.GetValue $MEMTABLE$
MemoryTable.SetValue $MEMTABLE$, VALUE
MemoryTable.AppendValue $MEMTABLE$, VALUE
MemoryTable.ClearValue $MEMTABLE$
MemoryTable.ReplaceValue $MEMTABLE$, FINDVALUE , REPLACEVALUE

Success Criteria Rules: XSD Schema Validation

The Validate XSD Schema feature allows you to perform XSD schema validation of the document against the Project schema to ensure the structure and data conforms to the expectations of the schema definitions. For WSDL projects the schema(s) are defined by the WSDL. For custom projects, you can specify the Schema by clicking on the documents node under configuration and choosing the load schema option.



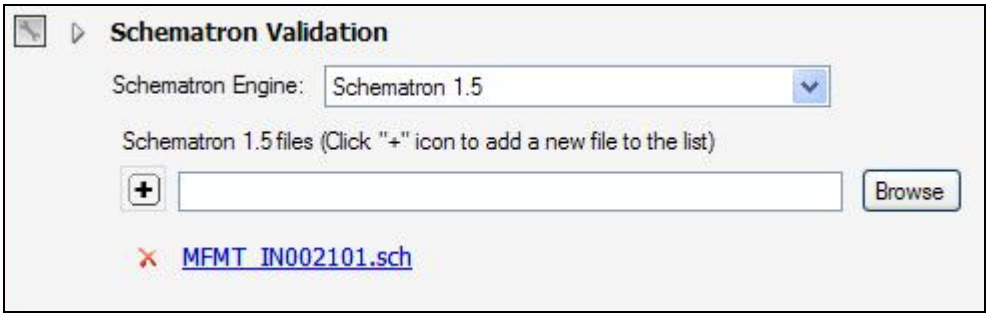
For schema validation, you can choose to ignore SOAP faults since often a SOAP fault structure is not included in the schema, but SOAP faults is the appropriate provision per SOAP specification to use for error response. Thus, from a structure standpoint a SOAP fault can be considered valid. To allow for this, check the “Allow SOAP Faults” checkbox.

Note: This feature requires a valid XSD schema. If the XSD Schema is not able to be compiled due to errors in the schema, incomplete references, or invalid schema definition structure, the Validate XSD schema feature will not be able to properly validate the messages.

Success Criteria Rules: Schematron Validation

The Schematron Validation success criteria rule allows you to analyze the document against a set of Schematron definitions, which provide a language for making assertions about the presence or absence of patterns in XML documents. CloudPort supports two standards for Schematron: Schematron 1.5 and ISO Schematron. Based on the selection of this version, the appropriate Schematron engine will be used to evaluate the Schematron definition files against the inbound document.

For Schematron 1.5 version engine validation, select the set of Schematron files to be used to validate the incoming document. To add new files to the list, use the browse button to browse for the file, then click the "+" icon to add the file to the list.



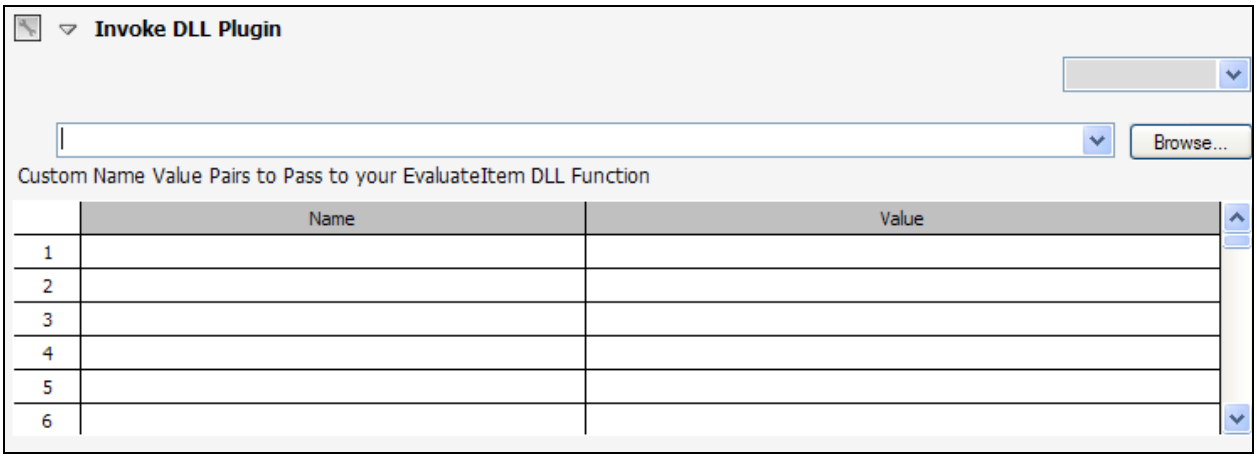
To use the ISO Schematron validation engine, first select "ISO Schematron" from the engine list. On the dialog that appears, choose the primary Schematron file and be sure to add all include files. You will be prompted to add the detected include files once you press the "+" icon to add the primary Schematron file).

Tip: To maximize screen space when configuring rules, click on the rule title to open the rule configurable in a separate popup window.

Success Criteria Rules: Invoke DLL Plugin

You can build you own evaluation functions using the extensible CloudPort DLL plug-in interface. This interface allows you to build a DLL which implements the **EvaluateItem** function signature as documented in the interface document ICloudPortPlugin.vb found in the plugins directory under the installation directory. You can write the plug-in in any language since CloudPort invokes the DLL using reflection to determine whether the function signature has been implemented. If the function signature is found, the function **EvaluateItem** function will be invoked.

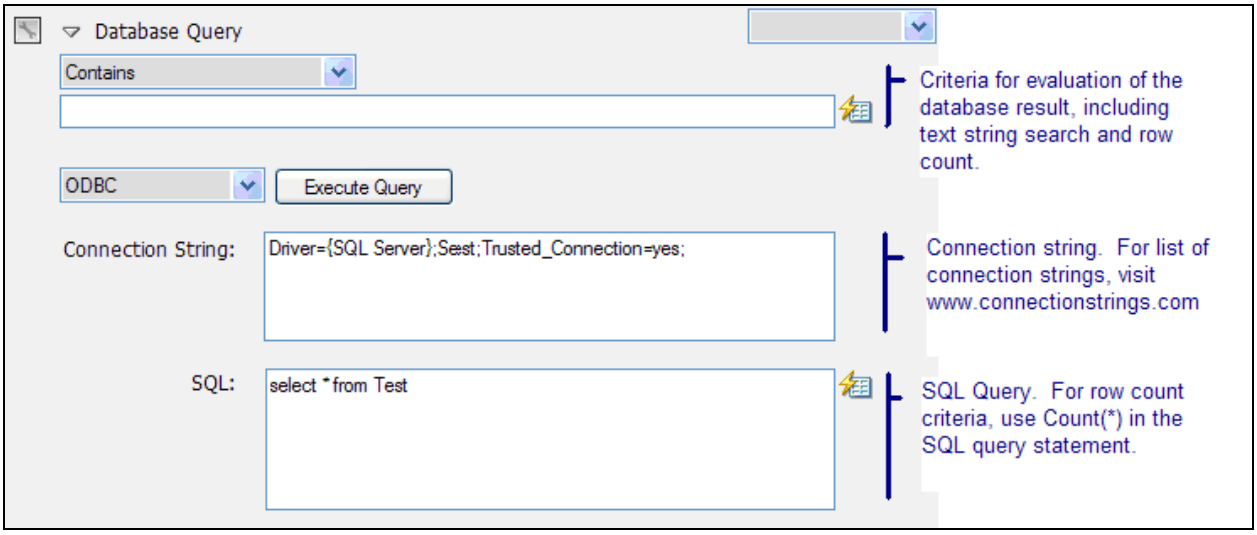
This function will be passed the request data, response data, and any variable substitutions in-scope for the current test iteration. You can also pass name/value pairs to the selected DLL by entering in the table provided.



Tip: To maximize screen space when configuring rules, click on the rule title to open the rule configurable in a separate popup window.

Success Criteria Rules: Database Query

You can run any defined query against a target database as a success criteria event. Any SQL query, such as SELECT, INSERT, DETELE, DROP TABLE, etc can be run and the results of the query analyzed for the criteria (row count, value match, etc).



When creating a database query, just as with other success criteria rules, you can use variables from the test within the SQL query itself, or as part of the criteria.

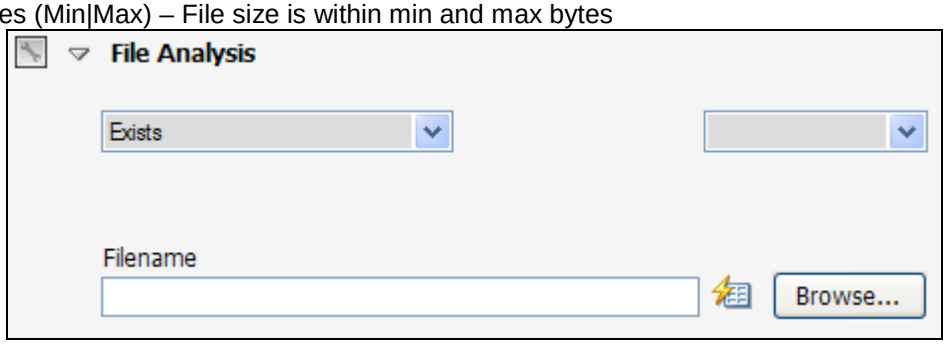
For connection strings to use for access to any ODBC compliant database, visit www.connectionstrings.com

Tip: To maximize screen space when configuring rules, click on the rule title to open the rule configurable in a separate popup window.

Success Criteria Rules: File Analysis

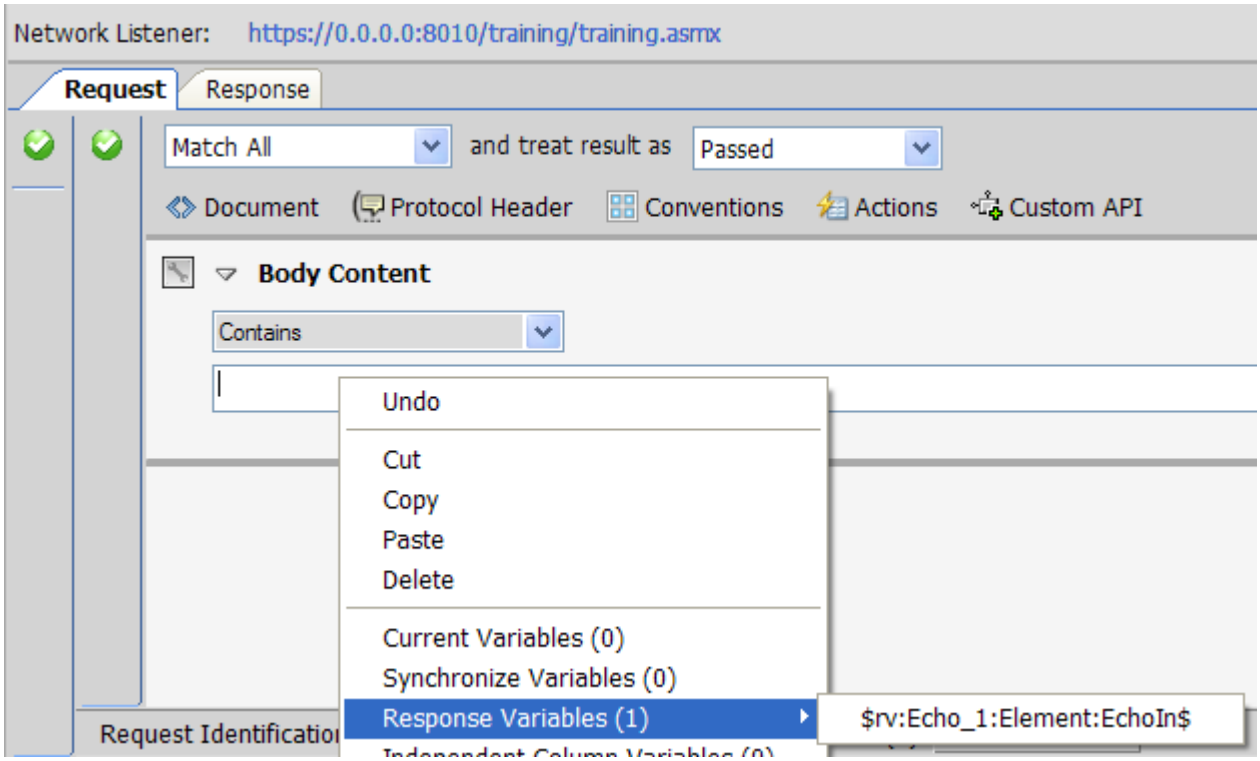
You can perform file analysis as a success criteria item. File analysis criteria include:

Exists – Referenced file exists
Does Not Exist – Referenced file does not exist
Contains – Referenced file contains a string match of the referenced data
Does Not Contain - Referenced file does not contain a string match of the referenced data
Regular Expression – Referenced file contents matches the referenced data regular expression
File Bytes (Min|Max) – File size is within min and max bytes



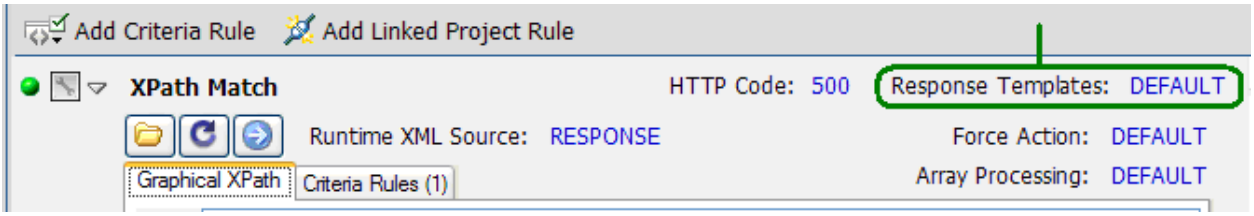
Success Criteria Dynamic Variables

You can dynamically parameterize the Success Criteria. This allows you to pull the data used to evaluate success and failure dynamically from Runtime Variables values used in the request. To use a runtime variable in the success criteria, right-click in the field and choose from the variable shown.



DYNAMIC RESPONSE TEMPLATES

The CloudPort Success Criteria features enable association of the response template used to send the response back to the client under conditions established by each matching criteria rule. The response templates can contain variables, and the templates can also be configured to use optional XSLT processing prior to sending the information back to the client.

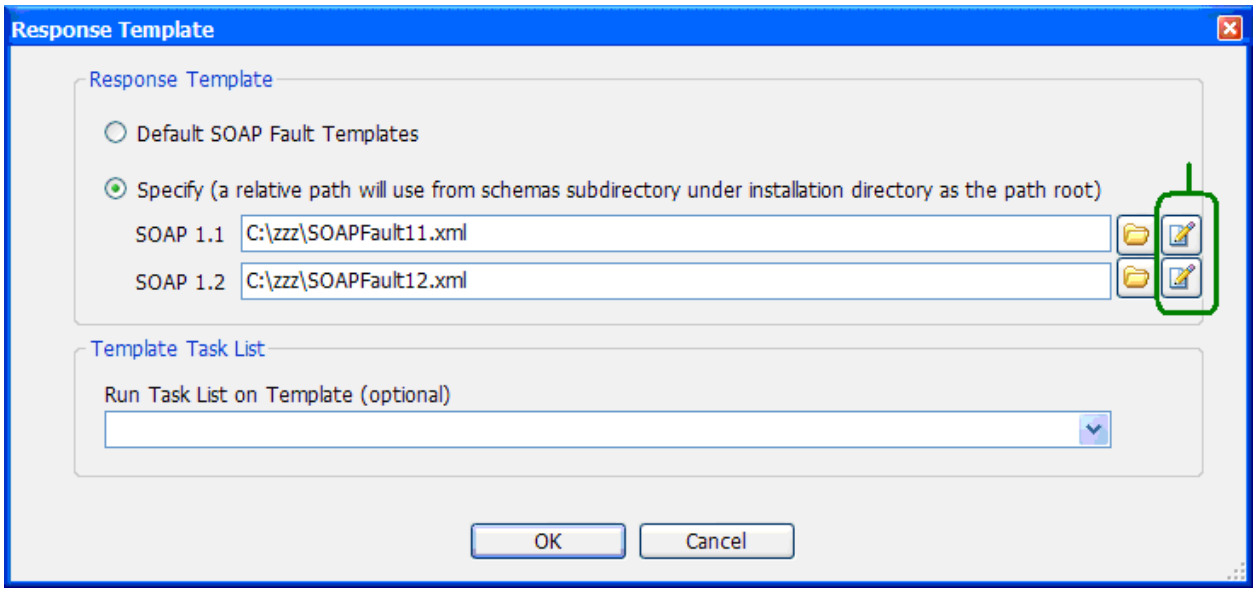


Response Templates

Each criteria rule allows the independent association of the rule trigger with a SOAP 1.1 and a SOAP 1.2 fault template. If using XML or REST test case types, the default template will always be the SOAP 1.1 template. If SOAP is being used, then the template response will be based on the SOAP version used in the client request.

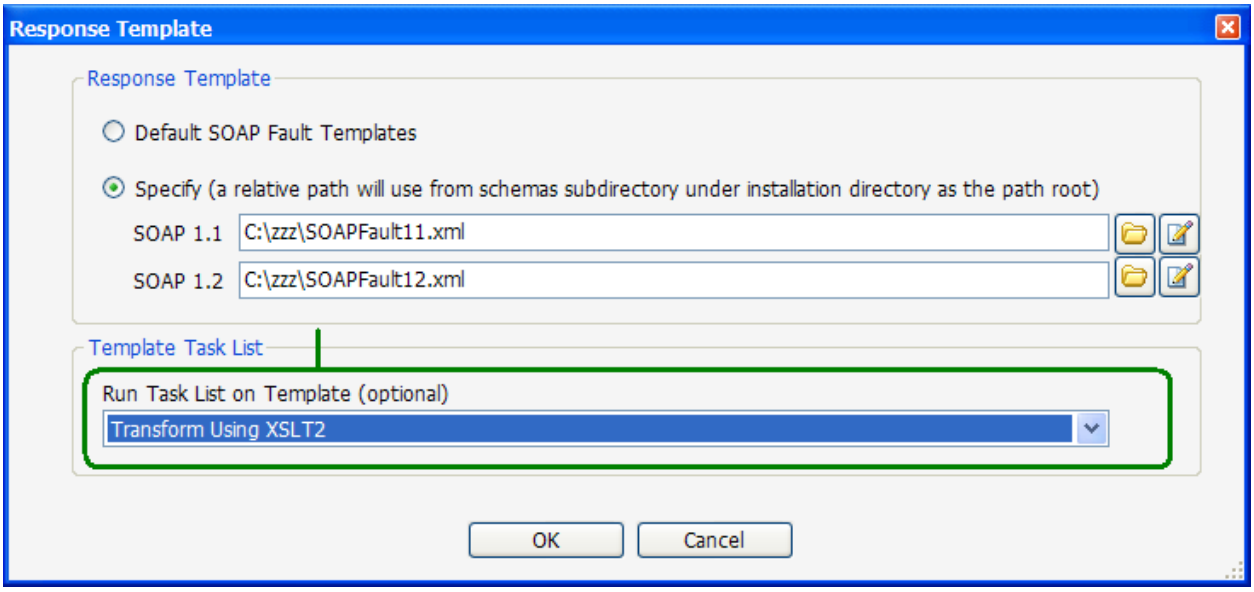
Variable Awareness

Templates are variable aware and will resolve any detected Context Functions, Runtime Variables, and Global Variables prior to sending the template data back to the client.



Task List Processing

The configured response template can be processed against a task list group by selecting a task list group from the list provided. Task list groups are defined from the project Policy Settings -> Task Groups menu.

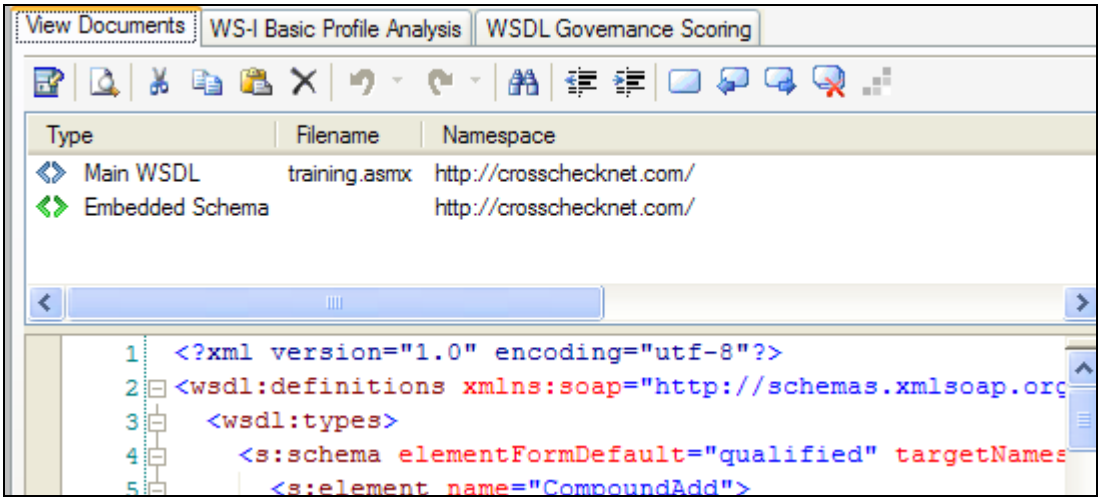


WSDL ANALYSIS AND SCORING

Subtopics in this section include:

- Document Analysis
- WSDL Analysis using WS-I Profiles
- WSDL Scoring Rules
- WSDL Score

Document Analysis

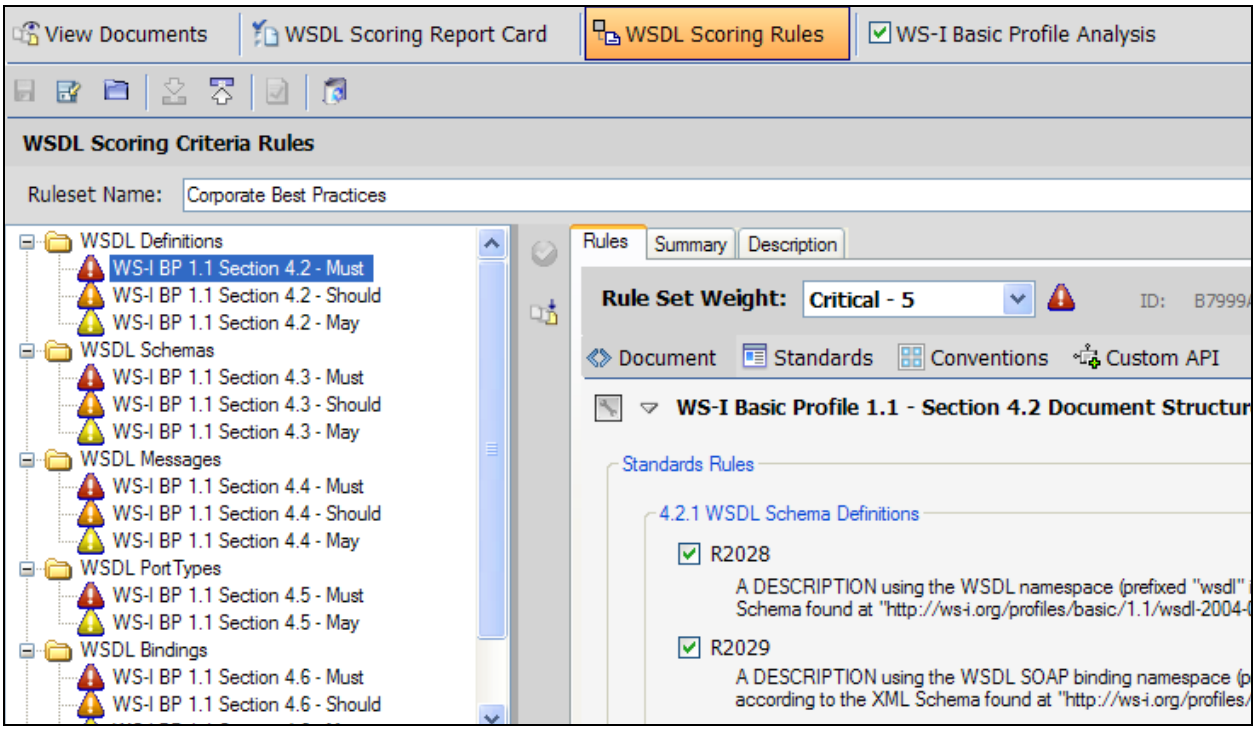


Design Time WS-I Basic Profile 1.1 WSDL Analysis

Crosscheck Networks is a member of the WS-I Organization and provides integrated design time WSDL analysis scanning using the WS-I assertion engine. WSDL analysis can be performed using 1 of 3 standard WS-I profiles. This analysis looks for violations based on the WS-I profile rules which evaluate against the structure and content of the WSDL document. CloudPort extends the WS-I assertion engine by also providing failed assertion WSDL section highlighting and an assertion result summary.

WSDL Scoring

CloudPort provides rules combining industry standard WS-I Basic Profile assertions with custom defined rules to score each component of the WSDL including: WSDL Definitions, WSDL Schemas, WSDL Messages, WSDL Port Types, WSDL Bindings, and WSDL Services.

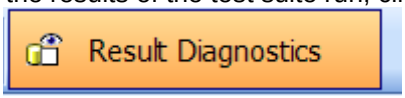


LOGGING AND REPORTING

Test suites store results in serialized XML files called log files. Each run mode stores files to the specified active directory. Viewing log files from test suite runs will provide access to report diagnostics specified to the active run mode. For example, when viewing logs and reports in QA mode, the reports are success based to assess the functional success or failure of Simulators. In Performance mode, the reports provide analysis criteria for performance profile measurement statistics. In Compliance mode, the reports show which messages violated the WS-I Basic Profile 1.1 SOAP Messaging requirements. In Vulnerability mode, the report shows the AVD criteria that matched the response data set and the risk posture based on the matched criteria items.

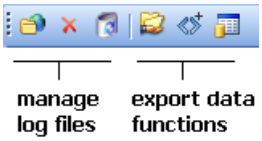
Subtopics in this section include:

To view the results of the test suite run, click the navigation icon to go to the Logging and Reporting area.



The log files appear on the left pane organized by date. To view the information in a log file, select the entry. Contents will appear on the right screen based on the type of log file and run mode that was used to create the result set. Within each run mode there is a set of reports that can be generated from the log results.

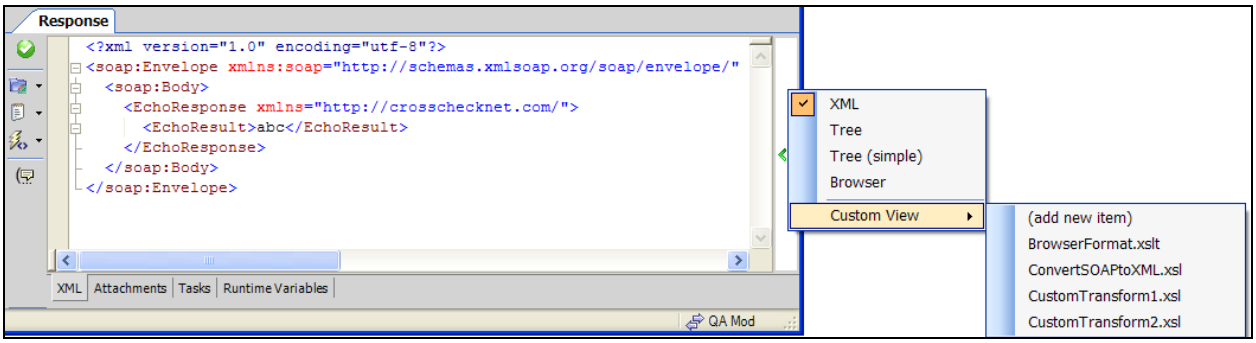
Result Log Toolbar



The log toolbar provides options for changing the active log folder for the current run mode, deleting the log file, or deleting a set of log files. There are also several export functions that allow you to extract all request and response data as individual files into a directory, export the log files as a normalized (no hashes) result set, and generate an HTML baseline XML diff report (exclusive to QA mode).

Log Message View Perspectives

There are several different ways to view the request and response transaction data. You can view raw xml, tree view, simple tree view, browser, or define and load your own custom XSLT transform files to provide customized views and formats of data.. To access the view perspectives, click on the icon to the right of the response display. The view perspectives are explained in more detail below.

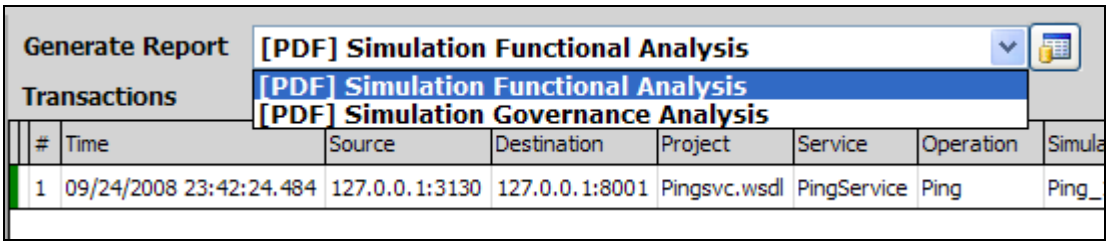


XML:	Used to edit and view raw data
Tree:	Used to view graphical tree of XML (read only)
Tree (simple):	Omits Namespaces and Attributes. Shows only elements with values (read only)
Browser:	Renders XML similar to Internet Explorer (read only)
Custom View	Allows selection of custom XSLT to format view of response data as HTML

For the Custom View option, selecting “add new item” will allow browsing for XSLT files to add to the list of available custom view options. These files can also be copied directly to the lib/xslt directory under the installation directory which is scanned at start-up.

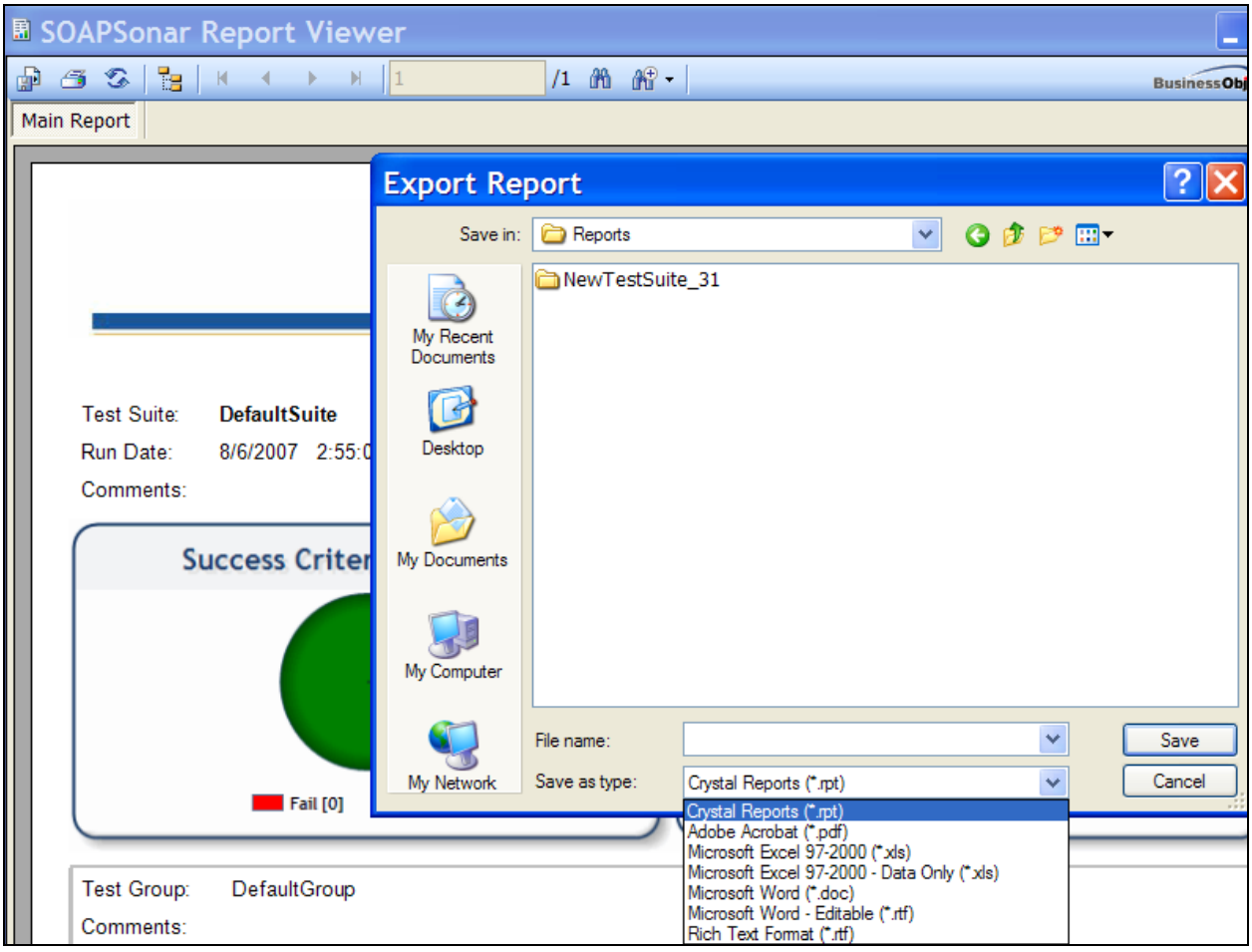
Generating Reports

When viewing log results reports can be generated in the report viewer. CloudPort uses Crystal Reports for rendering of reports in the report viewer which provide navigation among report pages as well as export formats for report data. To generate a report, click on the reports to select the report type and then press the Generate Report button to create the selected report and open the report viewer.



Report Export Formats

There are several formats for exporting reports including .pdf, .xls, .doc, .rtf, and .rpt. To export the report click on the top left button in the report viewer and then select the type of document to export the report to.



Exporting Raw Request and Response Data to File

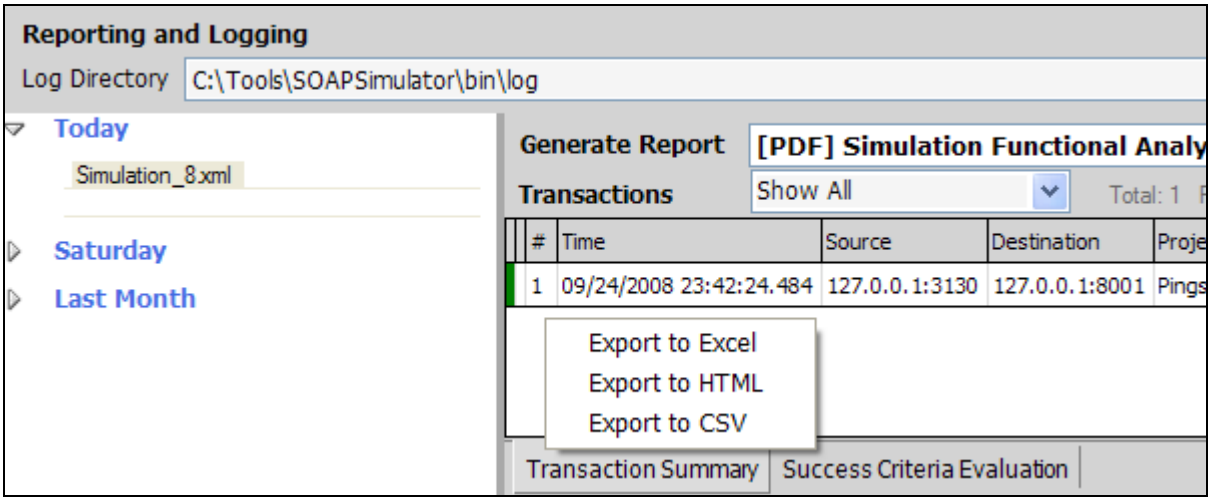
This option will extract each request and response message that was sent and received into the selected directory. A file will be created for each request and each response. You can choose whether to include the HTTP header information that was sent and received in the output files. Select this option from the log toolbar, or right-click on the log filename.

Exporting Raw Log Results to Normalized XML

By default, the log file results are stored in relational XML format to with hash mapping indexes. To export the result data in normalized (flat) format, select this option from the log toolbar, or right-click on the log filename. The exported results will match the current entries displayed in the summary grid, which are based on the Filter settings ("Show All", "Show Matching", and "Show Non-Matching").

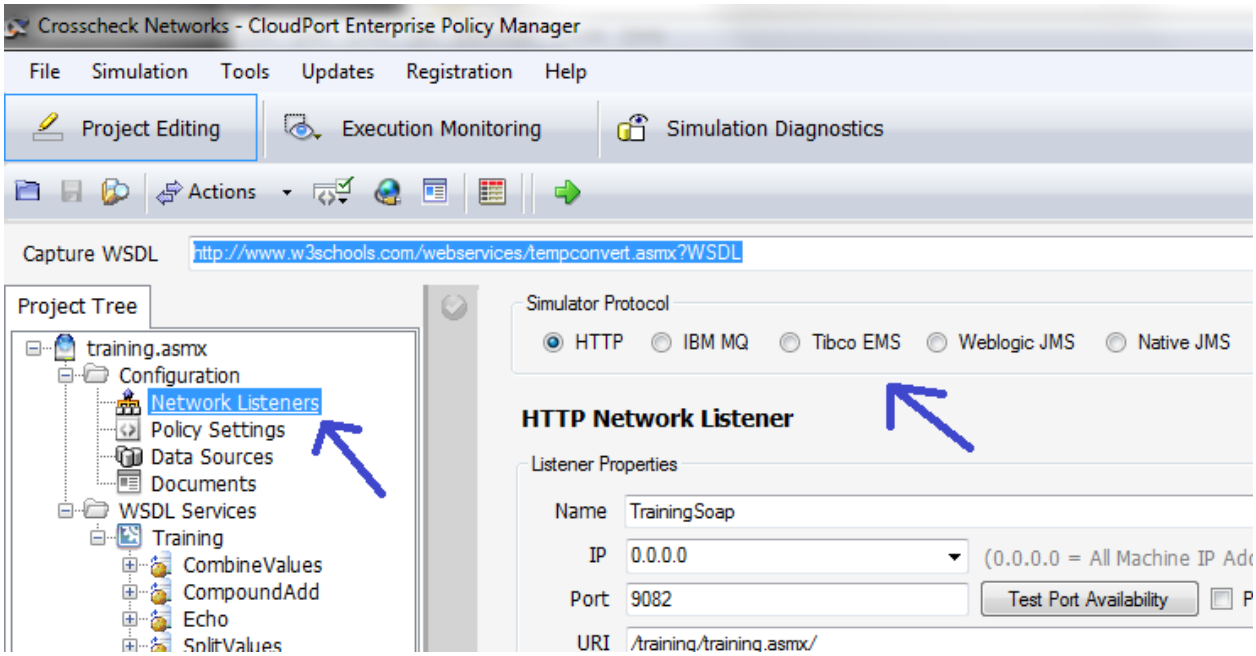
Exporting Log View to Excel or HTML Table

To export the summary information to an HTML table or Excel file, right click on the log view area and select the Export to Excel or Export to HTML option



IBM MQ, Weblogic JMS, Tibco EMS, and Native JMS MESSAGE PROVIDERS

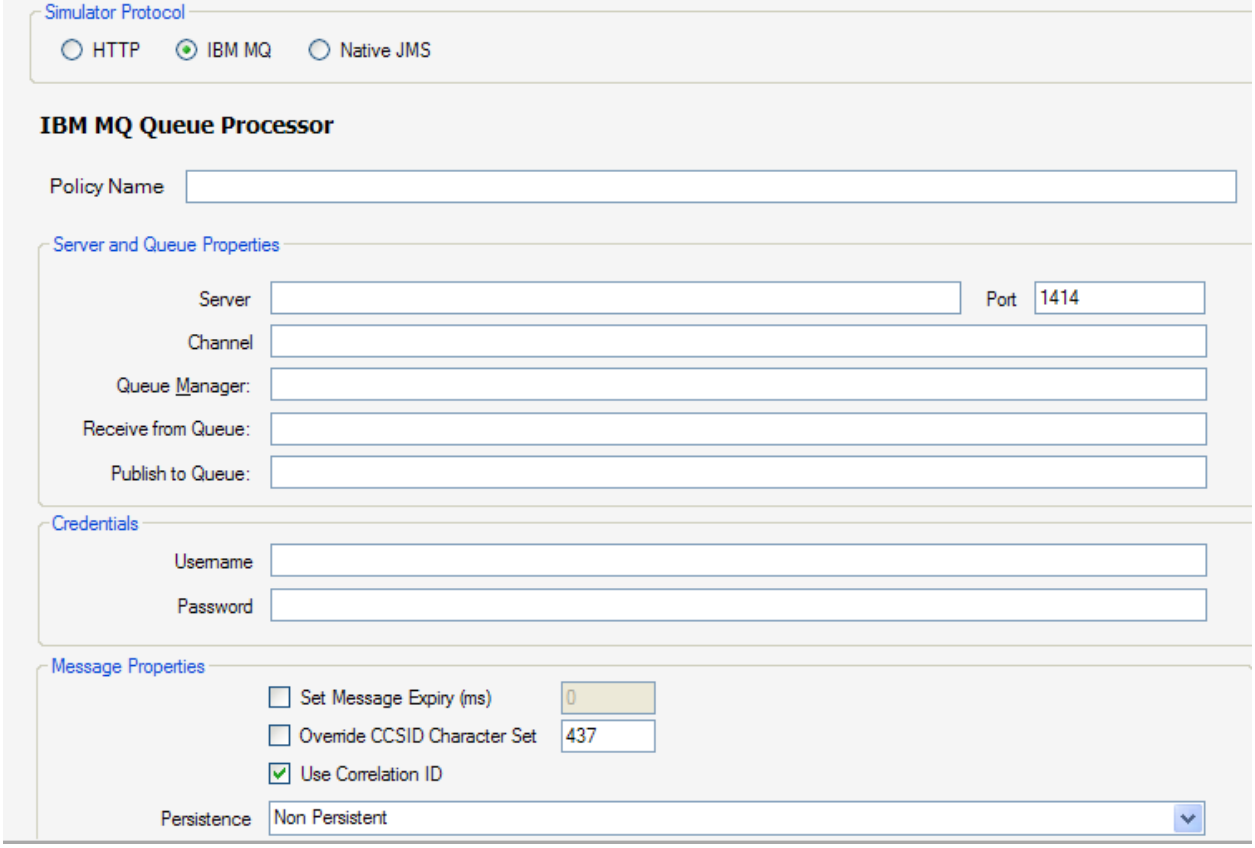
For IBM MQ, Weblogic JMS, Tibco EMS, or native JMS messaging, the policy configuration is located on the Message Providers section of the Configuration area.



MQ Message Provider

The MQ message provider allows for native MQ protocol testing using the IBM MQ Client libraries. In order to send MQ messages, you must first install the IBM MQ Client on the machine where CloudPort is running. The IBM MQ client is freely available from the IBM web site.

Once you have the IBM MQ Client installed, you can create MQ policies to send and receive messages from MQ queues. The settings for an MQ policy are described below.



Policy Name:
The name used to reference the policy from the URI field in a test case

Server:
The host name of the MQ Queue Manager

Channel:
The MQ Channel where the Queue Manager is defined

Queue Manager:
The name of the Queue Manager where the queues are defined

Receive from Queue;
The name of the Queue to receive messages from.

Publish to Queue:
The name of the Queue to send the processed message obtained from the receive queue

Read Timeout:
Amount of time to wait while listening for a response message on the receive Queue

Username:
If credentials are required, the username

Password:
If credentials are required, the password

Message Expiry:
The message property indicating whether the message has an expiry

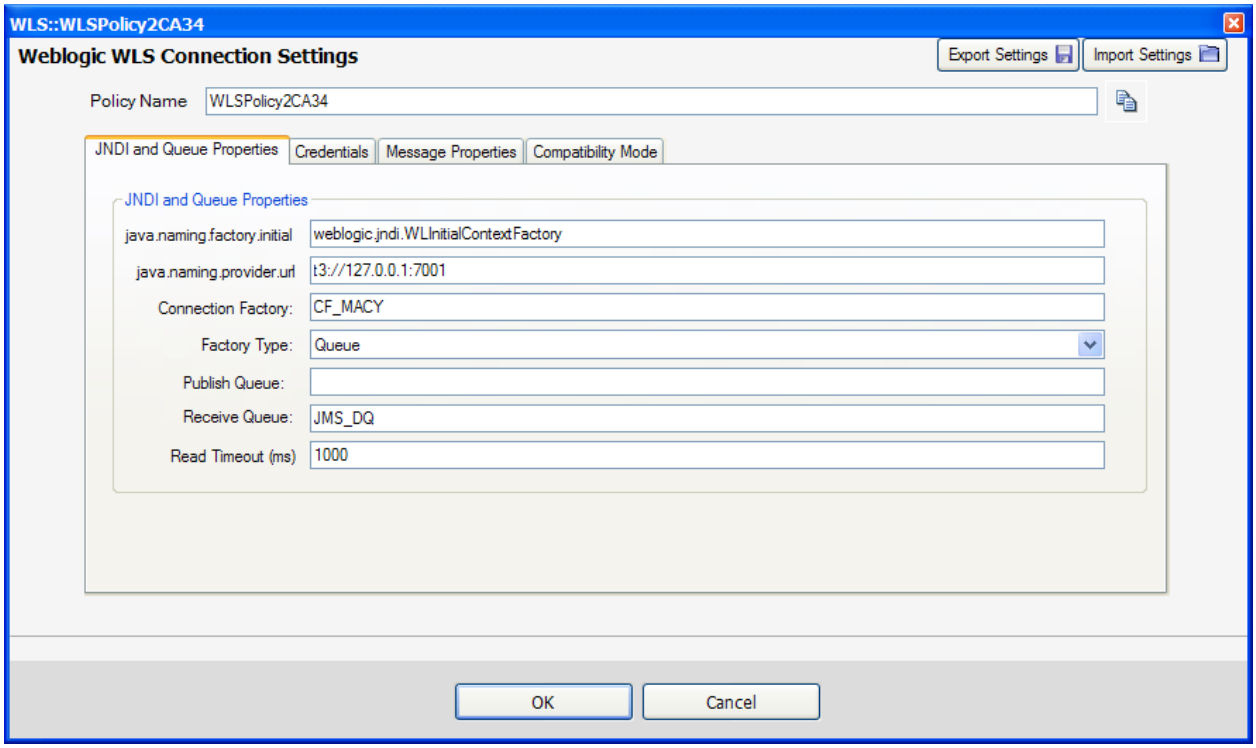
Use Correlation ID:
When set, expects the response message correlation ID to match the originating message ID

Persistence
The persistence setting for the published message

Oracle Weblogic JMS Message Provider (WLS)

The Oracle Weblogic JMS message provider allows for Weblogic JMS protocol testing using the native Weblogic API. In order to send WLS messages, you must first install the Weblogic .NET Client `WebLogic.Messaging.dll` on the machine where CloudPort is running. The Weblogic client is freely available from the Oracle web site, and can be obtained from a Weblogic installation.

Once you have the Weblogic Client installed, you can create WLS policies to send and receive messages from Weblogic queues. The settings for a Weblogic policy are described below.



Policy Name:

The name used to reference the policy from the URI field in a test case

Java.Naming.Factory.Initial:

This is the context factory to be used to establish the InitialContext() call

Java.Naming.Provider.Url:

The endpoint listener which provides the JNDI information

ConnectionFactory:

The name of the Connection Factory class

Factory Type:

The type of messaging exchange to be performed

Publish Queue:

The name of the Queue to send message to. Used if 2-way or 1-way publish options are selected.

Receive Queue;

The name of the Queue to receive messages from. Used if 2-way or 1-way receive options are selected

Username

Username for JNDI Context bind and Queue access

Password

Password for JNDI Context bind and Queue access

Read Timeout:

Amount of time to wait while listening for a response message on the receive Queue

Message Expiry:

The message property indicating whether the message has an expiry

Use Correlation ID:

When set, expects the response message correlation ID to match the originating message ID

Persistence

The persistence setting for the published message

Format

Choice of JMS Map Message format or JMS Text Message format

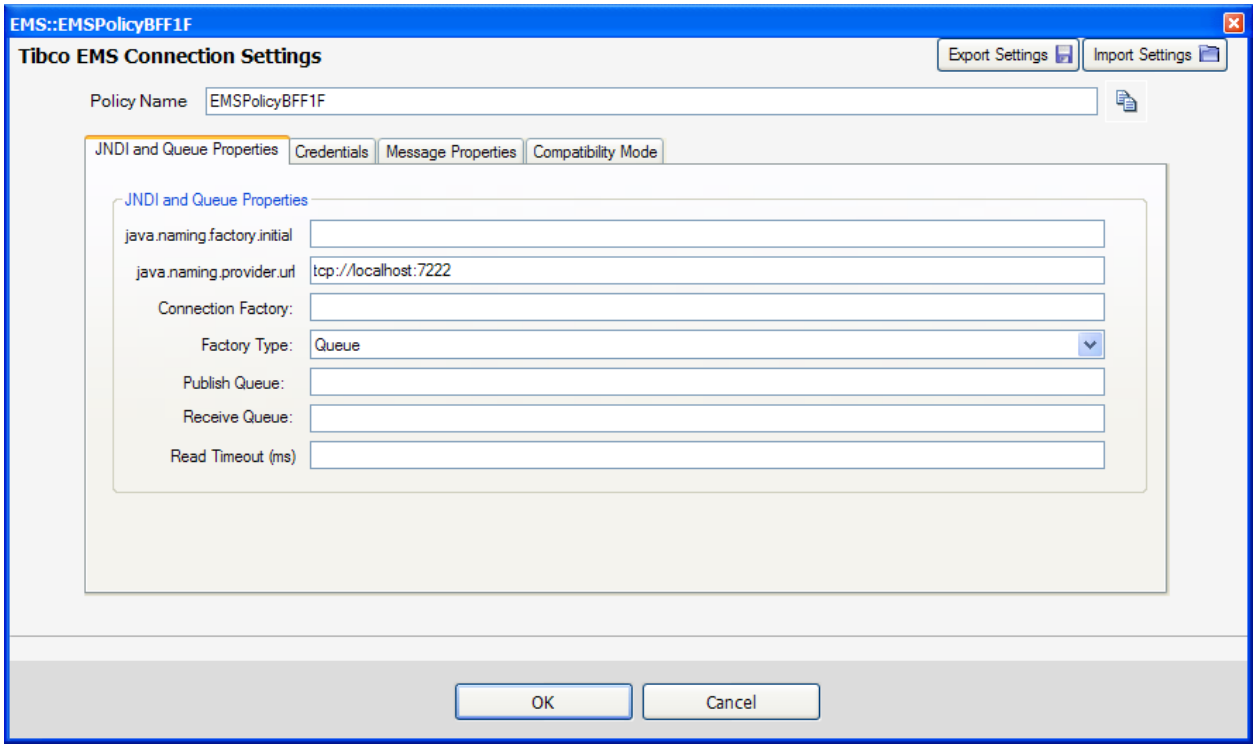
Compatibility Mode

Version of the API to invoke

Tibco EMS Message Provider

The Tibco EMS message provider allows for Tibco EMS protocol testing using the native Tiboc API. In order to send EMS messages, you must first install the Tibco .NET Client `TIBCO.EMS.dll` on the machine where CloudPort is running. The Tibco .NET client is freely available from the Tibco web site, and can be obtained from a Tibco EMS installation.

Once you have the Tibco .NET Client installed, you can create EMS policies to send and receive messages from Tibco EMS queues. The settings for a Tibco policy are described below.



Policy Name:

The name used to reference the policy from the URI field in a test case

Java.Naming.Factory.Initial:

This is the context factory to be used to establish the InitialContext() call

Java.Naming.Provider.Url:

The endpoint listener which provides the JNDI information

ConnectionFactory:

The name of the Connection Factory class

Factory Type:

The type of messaging exchange to be performed

Publish Queue:

The name of the Queue to send message to. Used if 2-way or 1-way publish options are selected.

Receive Queue;

The name of the Queue to receive messages from. Used if 2-way or 1-way receive options are selected

Username

Username for JNDI Context bind and Queue access

Password

Password for JNDI Context bind and Queue access

Read Timeout:

Amount of time to wait while listening for a response message on the receive Queue

Message Expiry:

The message property indicating whether the message has an expiry

Use Correlation ID:

When set, expects the response message correlation ID to match the originating message ID

Persistence

The persistence setting for the published message

Format

Choice of JMS Map Message format or JMS Text Message format

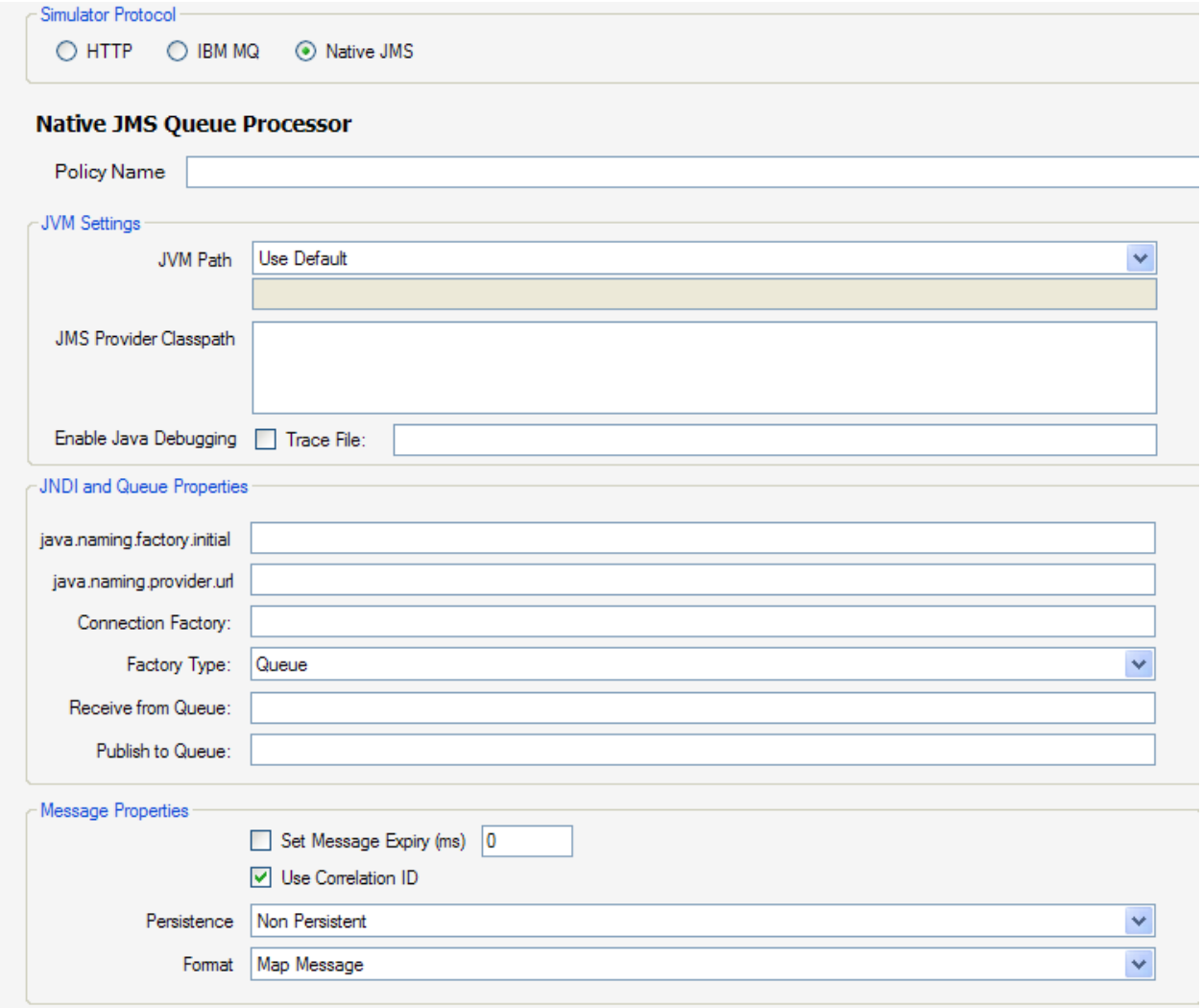
Compatibility Mode

Version of the API to invoke

JMS Message Provider

The JMS message provider allows for native JMS protocol testing using the JMS provider of your choice. In order to send JMS messages for your selected JMS provider, you must first install the provider JAR files on the machine where CloudPort is running. The JMS message provider uses JNDI to define the classes and connection information.

The settings for configuring a JMS message provider policy are defined below.



Policy Name:

The name used to reference the policy from the URI field in a test case

JVM Path:

Choose “Use Default” to use the active JVM based on the JAVA_HOME environment variable. Otherwise, choose “Specify” and enter the path to the jvm.dll directly.

JVM Provider Classpath:

The classpath for the chosen JMS provider JAR files. If multiple JAR files are required, separate them with a “;” delimiter. i.e. a.jar; b.jar; c.jar.

Enable Java Debugging:
Allows debug tracing of the Java Virtual Machine to diagnose connection or initialization issues

Java.Naming.Factory.Initial:
This is the context factory to be used to establish the InitialContext() call

Java.Naming.Provider.Url:
The endpoint listener which provides the JNDI information

ConnectionFactory:
The name of the Connection Factory class

Factory Type:
The type of messaging exchange to be performed

Receive from Queue;
The name of the Queue to receive messages from.

Publish to Queue:
The name of the Queue to send the processed message obtained from the receive queue

Read Timeout:
Amount of time to wait while listening for a response message on the receive Queue

Message Expiry:
The message property indicating whether the message has an expiry

Use Correlation ID:
When set, expects the response message correlation ID to match the originating message ID

Persistence
The persistence setting for the published message

Format
Choice of JMS Map Message format or JMS Text Message format

EXTENSIBLE PLUGIN API

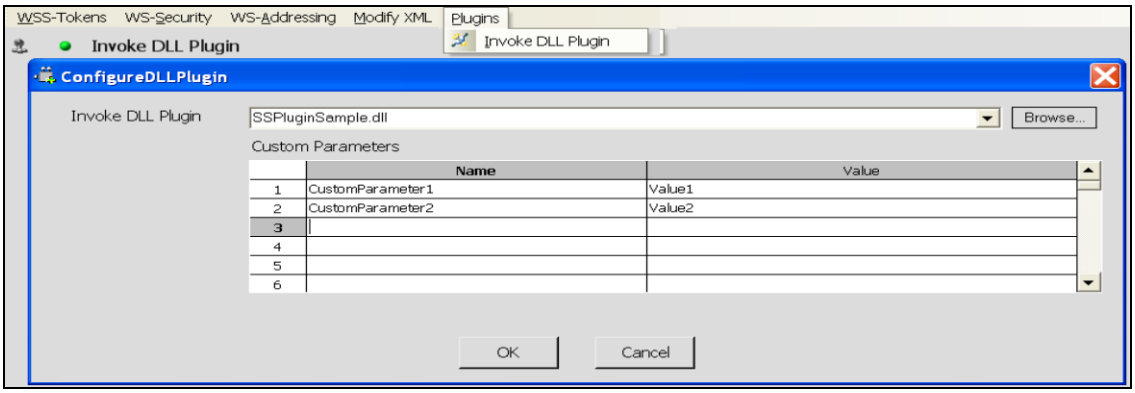
CloudPort provides a plug-in interface which allows you to write your own DLL plug-ins to be invoked during the request, response, and success criteria evaluation events. An interface document is provided in the installation subdirectory called **plugins**. The interface DLL is called CloudPort.IPlugin.dll and describes each function signature that must be implemented to be called for the events. The functions are documented in the interface document provided in the plugins subdirectory and are in VB.NET and C# format. Once you have created a DLL which implements the interfaces specified in CloudPort.IPlugin.dll, you can configure the functions to be invoked for any of 3 events: Request Event, Response Event, and Evaluate Response event.

Request Event

On the Request Tasks tab, the Invoke DLL Plugin menu item allows you to select your DLL to be invoked for each request. The request event triggers the RunCustomRequestTask function. Values returned from your function for the Request and Request Header will be used by CloudPort for the current request and header to be sent to the specified endpoint.

```
public bool RunCustomRequestTask(ref string  Request, ref string RequestHeader , ref HashTable Variables);
```

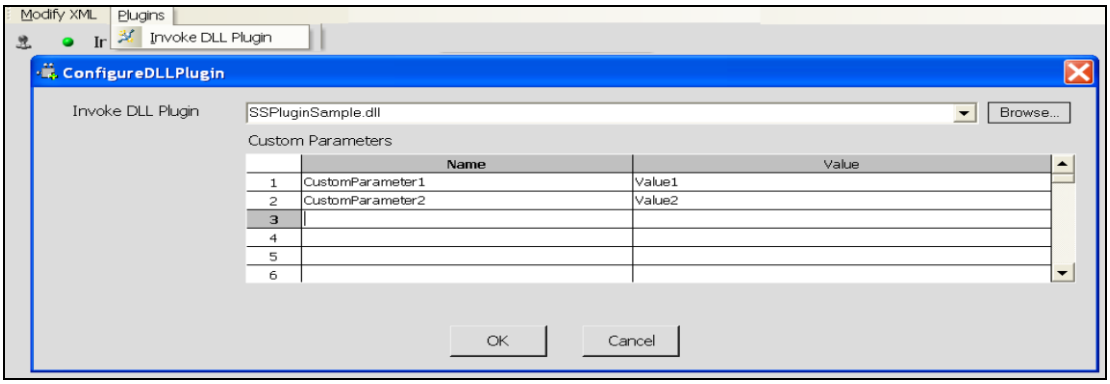
Custom parameters can be passed to your DLL function by specifying them in the table provided.



Response Event

On the Response Tasks tab, the Invoke DLL Plugin menu item allows you to select your DLL to be invoked for each response. The response event triggers the RunCustomResponseTask function. Values returned from your function for the Response and Response Header will be used by CloudPort for the success criteria evaluation and Runtime Variable capture.

```
public bool RunCustomResponseTask(ref string Response , ref string ResponseHeader, ref Hashtable Variables);
```

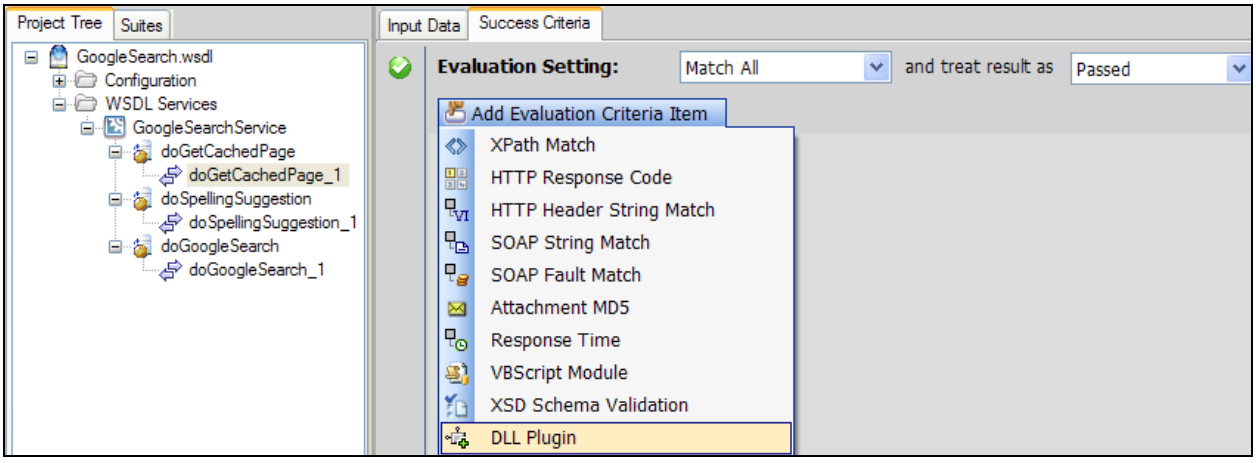


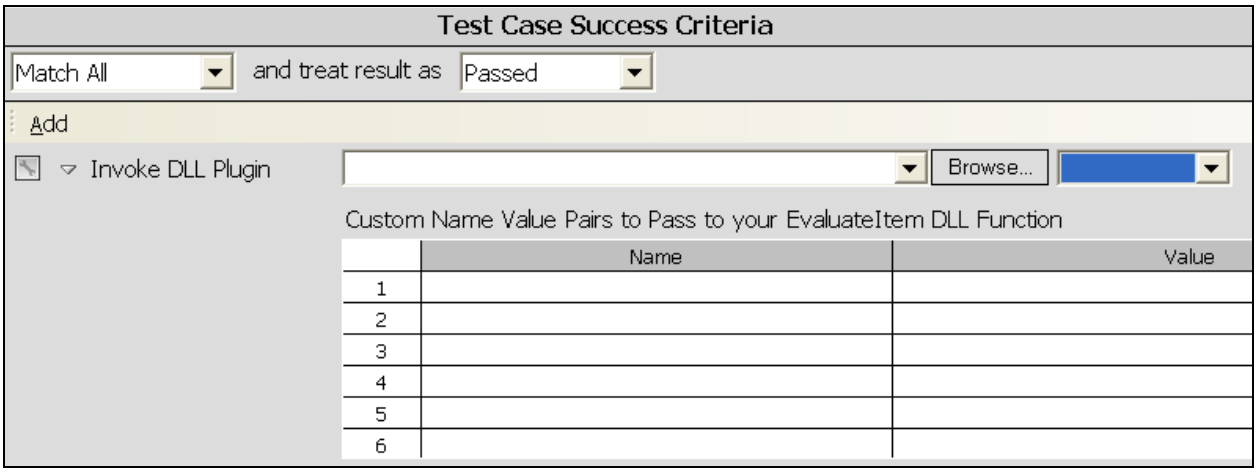
Evaluate Success Event

On the Success Criteria tab, the Invoke DLL Plugin menu item allows you to select your DLL plug-in to be invoked for success/failure assessment of the current response. The response event triggers the EvaluateItem function.

```
public bool EvaluateItem(ref string Request, ref string RequestHeader,
                         ref string Response, ref string ResponseHeader,
                         ref HashTable Variables, ref ArrayList Attachments,
                         ref string MatchString , ref string MatchContext ,
                         ref string MatchExpression , ref string MatchCriteria);
```

Custom parameters can be passed to your DLL function by specifying them in the table provided.




Test Case Success Criteria
Match All
and treat result as
Passed
Add
Invoke DLL Plugin
Browse...
Custom Name Value Pairs to Pass to your EvaluateItem DLL Function
Name
Value
1
2
3
4
5
6

TESTING SIMULATIONS

Crosscheck Networks SOAPSonar can be used to test the simulation rules. SOAPSonar is available as a free product download from http://www.crosschecknet.com/products/soapsonardetails_personal.php.

SOAPSonar provides simple, intuitive, and comprehensive testing for SOAP, XML, and REST based services over HTTP, HTTPs, MQ, and JMS protocols. The SOAPSonar testing framework is easy to deploy and provides testing modes for functional, performance, compliance and security testing.

Crosscheck Networks designed SOAPSonar from the ground-up as a comprehensive testing solution with features that can be utilized throughout all phases of the service development lifecycle. SOAPSonar is easy to deploy and requires no knowledge of SOAP, XML, or WSDL to be effective. Services can be tested within minutes, and more complex testing scenarios can be built including complex security, identity, automation, and performance.

Launch SOAPSonar Testing Client

You can launch SOAPSonar directly from CloudPort Runtime Player by going to the **Testing->Launch SOAPSonar Testing Client**.

Appendix A – Now() Context Function Date Formats

Supported date formats for the Now() context function to format the current date and time according to the format definitions within the parameters.

The following are examples of user-defined date and time formats for

`December 7, 1958, 8:50 PM, 35 seconds`:

Function	Displays
`$fn:Now(M/d/yy)$`	`12/7/58`
`$fn:Now(d-MMM)$`	`7-Dec`
`$fn:Now(d-MMMM-yy)$`	`7-December-58`
`$fn:Now(d MMMM)$`	`7 December`
`$fn:Now(MMMM yy)$`	`December 58`
`$fn:Now(hh:mm tt)$`	`08:50 PM`
`$fn:Now(h:mm:ss t)$`	`8:50:35 P`
`$fn:Now(H:mm)$`	`20:50`
`$fn:Now(H:mm:ss)$`	`20:50:35`
`$fn:Now(M/d/yyyy H:mm)$`	`12/7/1958 20:50`

List of user-defined data and time format options

Character	Description
(**:**)	Time separator. In some locales, other characters may be used to represent the time separator. The time separator separates hours, minutes, and seconds when time values are formatted. The actual character used as the time separator in formatted output is determined by your system's **LocaleID** value.
(***/***)	Date separator. In some locales, other characters may be used to represent the date separator. The date separator separates the day, month, and year when date values are formatted. The actual character used as the date separator in formatted output is determined by your locale.
(**%**)	Used to indicate that the following character should be read as a single-letter format without regard to any trailing letters. Also used to indicate that a single-letter format is read as a user-defined format. See below for further details
d	Displays the day as a number without a leading zero (for example, `1`). Use **%d** if this is the only character in your user-defined numeric format.
dd	Displays the day as a number with a leading zero (for example, `01`).
ddd	Displays the day as an abbreviation (for example, `Sun`).
dddd	Displays the day as a full name (for example, `Sunday`).
M	Displays the month as a number without a leading zero (for example, January is represented as `1`). Use **%M** if this is the only character in your user-defined numeric format.
MM	Displays the month as a number with a leading zero (for example, `01/12/01`).
MMM	Displays the month as an abbreviation (for example, `Jan`).
MMMM	Displays the month as a full month name (for example, `January`).
gg	Displays the period/era string (for example, `A.D.`)

h	Displays the hour as a number without leading zeros using the 12-hour clock (for example, `1:15:15 PM`). Use **%h** if this is the only character in your user-defined numeric format.
hh	Displays the hour as a number with leading zeros using the 12-hour clock (for example, `01:15:15 PM`).
H	Displays the hour as a number without leading zeros using the 24-hour clock (for example, `1:15:15`). Use **%H** if this is the only character in your user-defined numeric format.
HH	Displays the hour as a number with leading zeros using the 24-hour clock (for example, `01:15:15`).
m	Displays the minute as a number without leading zeros (for example, `12:1:15`). Use **%m** if this is the only character in your user-defined numeric format.
mm	Displays the minute as a number with leading zeros (for example, `12:01:15`).
s	Displays the second as a number without leading zeros (for example, `12:15:5`). Use **%s** if this is the only character in your user-defined numeric format.
ss	Displays the second as a number with leading zeros (for example, `12:15:05`).
F	Displays fractions of seconds. For example **ff** will display hundredths of seconds, whereas **ffff** will display ten-thousandths of seconds. You may use up to seven **f** symbols in your user-defined format. Use **%f** if this is the only character in your user-defined numeric format.
T	Uses the 12-hour clock and displays an uppercase `A` for any hour before noon; displays an uppercase `P` for any hour between noon and 11:59 P.M. Use **%t** if this is the only character in your user-defined numeric format.
tt	Uses the 12-hour clock and displays an uppercase `AM` with any hour before noon; displays an uppercase `PM` with any hour between noon and 11:59 P.M.
y	Displays the year number (0-9) without leading zeros. Use **%y** if this is the only character in your user-defined numeric format.
yy	Displays the year in two-digit numeric format with a leading zero, if applicable.
yyy	Displays the year in four digit numeric format.
yyyy	Displays the year in four digit numeric format.
z	Displays the timezone offset without a leading zero (for example, `-8`). Use **%z** if this is the only character in your user-defined numeric format.
zz	Displays the timezone offset with a leading zero (for example, `-08`)
zzz	Displays the full timezone offset (for example, `-08:00`)

Index