Last edited by **Ondřej Vašíček** 18 hours ago

1. Basics

Intro

We recommend using a REST client such as Postman to learn how to use the adapter's interface. Import the Tutorial collection into Postman to see actual rdf+xml HTTP requests. The collection uses variables to configure adapter address and ports (right click the collection -> edit -> variables)

Or use the Swagger UI provided by the adapter.

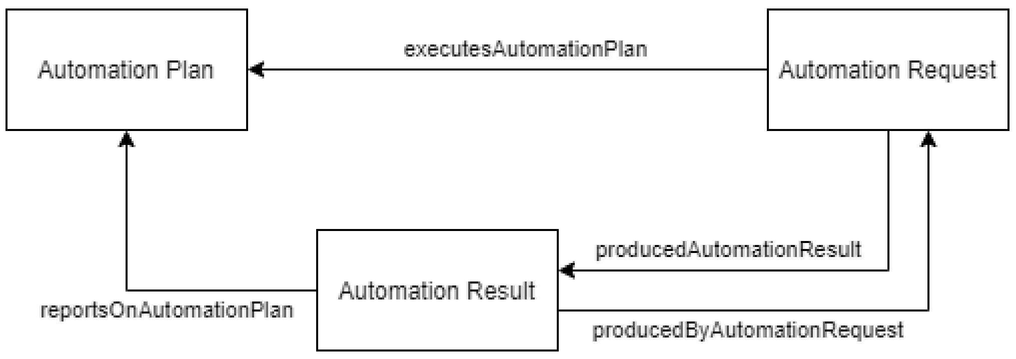
Basics

The adapter's interface is RESTful and models the OSLC Automation domain. A very basic summary of the Automation domain is that a server provides certain units of automation for clients using Automation Plans. Automation Plans define input parameters that can be used while requesting an execution of the unit of automation. A client then chooses one Automation Plan to execute and creates an Automation Request that references the selected Automation Plan and contains all the required input parameters. The server then executes the requested unit of automation and creates an Automation Result for it. The client needs to poll for the Automation Result and monitor its state. Automation Results contain contributions which can be anything from *stdout*, to files or links.

Types of requests:

- POST with an xml+rdf or json body to create resources
- GET with an accept header (html, xml+rdf, json) to read resources
- PUT to update resources
- DELETE to delete resources

Automation Resource connections



Usage

The adapter is divided into two separate adapters (different ports) - analysis and compilation.

A typical use case:

0. build using *build.sh or .ps1*, configure, and run using *run_all.sh or .ps1* (See Running)
1. deploy an SUT to the server using the compilation adapter
   - POST an Automation Request to the compilation adapter Request, Response
2. check the SUT URI
   - GET the Automation Result identified by the *producedAutomationResult* property Request, Response
3. use the analysis adapter to execute analysis on the SUT
   - POST an Automation Request to the analysis adapter referencing the *createdSUT* URI Request, Response
4. poll for the analysis result
   - GET the Automation Result identified by the *producedAutomationResult* property Request, Response

- See the the Tutorial collection for more detail

Compilation adapter

- Has a single Automation Plan - *{{compilation_address}}/compilation/services/resources/automationPlans/0*

Parameter definitions (input parameters):

- **launchCommand** - Command that will be executed by the Analysis adapter when executing the SUT. Optional parameter.
- **buildCommand** - Command executed by the Compilation adapter during SUT creation. Optional parameter.

- **compile** - A boolean flag that controls whether the SUT should be compiled using the buildCommand. Optional parameter with default value *true*.
- **unpackZip** - Value *true* means that the SUT file should be unzipped. Optional parameter.
- **SUT fetch method** - exactly one of the below parameters is to be used
  - sourceUrl - URL for direct download of the SUT
  - sourceGit - GIT repository to clone
  - sourceBase64 - base64 encoded SUT
  - sourceFilePath - File system path

Contributions (output resources):

- **Fetching Output** - output of the SUT fetching process
- **executionTime** - total duration of the compilation in milliseconds
- **statusMessage** - progress messages from the adapter
- **returnCode** - exit code of the compilation
- **stdout** - stdout of the compilation
- **stderr** - stderr of the compilation

Analysis adapter

- Has an Automation Plan for each analysis tool based on your configuration - *{{analysis_address}}/analysis/services/resources/automationPlans/**

Parameter definitions overview (See Analysis Tool Definition for details)

- **User defined tool command line parameters**
  - Each tool has its own input parameters for the command line (see the actual Automation Plans. These are entirely configured by the user based on what tool is the adapter being used for.
  - These parameters use the fit:commandlinePosition property to tell the adapter where to place the parameters value when launching the tool.
  - The simplest version is just one parameter that contains the whole command line arguments string.
- **Common input parameters**
  - **SUT** - URI of the SUT to analyse which was created by the compilation adapter. Required parameter as it determines the directory in which the tool will be executed.
  - **outputFileRegex** - Regex used to match new/modified files created by the analysis execution and add them as contributions. Optional (default "match none").
  - **zipOutputs** - All file contributions matched by the outputFileRegex will be ZIPed into a new contribution. Optional (default false).
  - **timeout** - Optional (default 0 ~ no timeout).
  - **outputFilter** - Used to select an output filter to process output contributions. Lists available filters through allowedValue properties. Optional with default value "default".
  - **confFile** - Allows a configuration file to be created in the SUT directory just before executing analysis. This parameter can be used multiple times to create multiple conf files.
  - **confDir** - Similar to confFile by accepts a base64 encoded ZIP of a directory full of configuration files. The expected value format is "path/to/place/dir\nbase64"
  - **beforeCommand** - A command to run just before executing analysis.
  - **afterCommand** - A command to run just after executing analysis.
  - **envVariable** - Environment variable to set for execution (analysis, and before and after commands). Can be used multiple times to specify multiple variables. Expected value "variable_name\nvariable_value"
- **Special input parameters** - Special parameter definitions that are not supposed to be supplied by the client. (used by the universalness of the adapter)
  - **launchSUT** - Instructs the analysis adapter to insert the SUT launch command at the specified command line position.
  - **SUTbuildCommand** - Instructs the analysis adapter to insert the SUT build command at the specified command line position.

Contributions (output resources)

- **default Contributions**:
  - **executionTime** - total duration of the analysisin milliseconds
  - **statusMessage** - progress messages from the adapter
  - **returnCode** - exit code of the compilation
  - **stdout** - stdout of the analysis
  - **stderr** - stderr of the analysis
  - ***filename*** - any number of contributions representing files produced or modified during analysis. Will have a fit:filePath property set. Contribution containing a filePath can be directly downloaded or uploaded as an octet-stream using its URI.
- **custom Contributions**

  - contributions produced by the analysis can be modified by user defined plugin filters (see Plugin Output Filters for details)