Validate Mediator

You can use the Validate mediator to validate XML and JSON messages.

Validating XML messages
Validating JSON messages

The Validate mediator validates XML messages against a specified schema. You can specify an XPath to extract and validate a specific part of the message. Otherwise, the mediator validates the first child of the SOAP body of the current message.

Syntax

```
<validate [source="xpath"]>
    <property name="validation-feature-id" value="true|false"/>
    <schema key="string"/>
    <on-fail>
        mediator+
    </on-fail>
</validate>
```

UI Configuration

The Validate mediator is a content aware mediator.

A Fault mediator should be added as a child to the Validate mediator in order specify the fault sequence to be followed if the validation fails.
The mediator configuration can be divided into the following sections.

**Schema keys defined for Validate Mediator**

This section is used to specify the key to access the main schema based on which validation is carried out, as well as to specify the XML which needs to be validated. The parameters available in this section are as follows.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema keys defined for Validate Mediator</strong> table</td>
<td>The key for the schema location. It can be specified using one of the following methods.</td>
</tr>
<tr>
<td></td>
<td>• If the key is a static value, select <strong>Static Key</strong> from the list and enter a static key in the data field. This value should be pre-defined and saved as a resource in the <strong>Registry</strong>. Click either <strong>Configuration Registry</strong> or <strong>Governance Registry</strong> as relevant to select the required key from the resource tree.</td>
</tr>
<tr>
<td></td>
<td>• If the key is a dynamic value, Select <strong>Dynamic Key</strong> from the list and enter an expression to calculate the value in the data field.</td>
</tr>
<tr>
<td>Click <strong>Add Key</strong> to add a new schema key. Click <strong>Delete</strong> in the relevant row to delete a schema key.</td>
<td></td>
</tr>
</tbody>
</table>

**Tip**

You can click **NameSpaces** to add namespaces if you are providing an expression. Then the **Namespace Editor** panel would appear where you can provide any number of namespace prefixes and URLs used in the XPath expression.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source</strong></td>
<td>The XPath expression to extract the XML that needs to be validated. The Validate mediator validates the evaluation of this expression against the schema specified in the <strong>Schema keys defined for Validate Mediator</strong> table. If this is not specified, the validation is performed against the first child of the SOAP body of the current message.</td>
</tr>
</tbody>
</table>

**Tip**

You can click **NameSpaces** to add namespaces if you are providing an expression. Then the **Namespace Editor** panel would appear where you can provide any number of namespace prefixes and URLs used in the XPath expression.

**Features Defined for Validator Mediator**

This section is used to specify which features of the Validate mediator should be enabled and which should be disabled. The parameters available in this section are as follows.

Only the **FEATURE_SECURE_PROCESSING** feature is currently supported by the validator.
<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Name</td>
<td>The name of the feature.</td>
</tr>
<tr>
<td>Value</td>
<td>Click True to enable the feature, or click False to disable the feature.</td>
</tr>
<tr>
<td>Action</td>
<td>Click Delete in the relevant row to delete a feature.</td>
</tr>
</tbody>
</table>

Resources of the Validate Mediator

A resource in the Validate mediator configuration enables you to import a schema referenced within another schema. In order to access such a schema via a resource, the parent schema should be saved as a resource in the Registry. The parameters available in this section are as follows.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>The location of the schema to be imported. The value entered here should be equal to the value of the schema location attribute within the relevant <code>&lt;xsd:import&gt;</code> element in the parent schema.</td>
</tr>
<tr>
<td>Key</td>
<td>The key to access the parent schema saved in the Registry. Click either Configuration Registry or Governance Registry as relevant to select the key from the resource tree.</td>
</tr>
</tbody>
</table>

Note

You can also configure the Mediator using XML. Click switch to source view in the Mediator window.

Examples

Example 1 - Basic configuration

In this example, the required schema for validating messages going through the validate mediator is given as a registry key, `schema\sample.xsd`. No source attribute is specified, and therefore the schema will be used to validate the first child of the SOAP body. The mediation logic to follow if the validation fails is defined within the `on-fail` element. In this example, the Fault Mediator creates a SOAP fault to be sent back to the party which sent the message.
Example 2 - Validate mediator with resources

In this example, the following schema named `08MockServiceSchema` is saved in the Registry. This schema is located in `MockDataTypes.xsd`. A reference is made within this schema to another schema named `08SOAPFaults`, which is located in `SOAPFaults.xsd`.

```xml
<xsd:import namespace= "http://samples.synapse.com/08MockServiceSchema" schemalocation="MockDataTypes.xsd">
  <xsd:import namespace= "http://samples.synapse.com/08SOAPFaults" schemalocation= "../Common/SOAPFaults.xsd">
</xsd:import>
```

The Validate mediator can be configured as follows.

```xml
<validate>
  <schema key="schema\sample.xsd"/>
  <on-fail>
    <makefault>
      <code value="tns:Receiver"
xmlns:tns="http://www.w3.org/2003/05/soap-envelope"/>
      <reason value="Invalid Request!!!"/>
    </makefault>
    <property name="RESPONSE" value="true"/>
    <header name="To" expression="get-property('ReplyTo')"/>
  </on-fail>
</validate>
```

The schema used by the validate mediator is `MockDataTypes.xsd`. In addition, a resource is used to import the `08SOAPFaults` schema, which is referred in the `08MockServiceSchema` schema. Note that the value `../Common/SOAPFaults.xsd` which is specified as the location for the schema to be imported is the same as the location specified for `08SOAPFaults` schema in the `08MockServiceSchema` configuration.

The on-fail sequence of this Validate mediator includes a Log mediator which is added as a child to the Validate.
mediator. This log mediator uses two properties to generate the error message `Validation failed` when the validation of a message against the schemas specified fails.

The Validate mediator validates JSON messages against a specified JSON schema. You can specify a JSONPath to extract and validate a specific part of the message. Otherwise, the mediator validates the complete content of the current message.

A Fault mediator or PayloadFactory mediator should be added as a child to the Validate mediator in order specify the fault sequence to be followed if the validation fails.

The Validate mediator is a content aware mediator.

**Syntax**

```xml
<validate [source="xpath"]>
  <schema key="string"/>+
  <on-fail>
    mediator+
  </on-fail>
</validate>
```

**UI Configuration**

The mediator configuration can be divided into the following sections.

**Schema keys defined for Validate Mediator**

```
<table>
<thead>
<tr>
<th>Static Key</th>
<th>Key</th>
<th>Configuration Registry</th>
<th>Governance Registry</th>
<th>Delete</th>
</tr>
</thead>
</table>
```

**Features defined for Validator Mediator**

```
Add feature
```

**Resources of the Validate mediator**

```
Location | Key | Configuration Registry | Governance Registry |
Add Resource
```

The mediator configuration can be divided into the following sections.

**Schema keys defined for Validate Mediator**
This section is used to specify the key to access the main schema based on which validation is carried out, as well as to specify the JSON message which needs to be validated. The parameters available in this section are as follows.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Schema keys defined for Validate Mediator** | The key for the schema location. It can be specified using one of the following methods.  
  - If the key is a static value, select Static Key from the list and enter a static key in the data field. This value should be pre-defined and saved as a resource in the Registry. Click either Configuration Registry or Governance Registry as relevant to select the required key from the resource tree.  
  - If the key is a dynamic value, Select Dynamic Key from the list and enter an expression to calculate the value in the data field.  

Click **Add Key** to add a new schema key. Click **Delete** in the relevant row to delete a schema key. |
| **Source** | The JSONPath expression to extract the JSON element that needs to be validated. The Validate mediator validates the evaluation of this expression against the schema specified in the **Schema keys defined for Validate Mediator** table. If this is not specified, the validation is performed against the whole body of the current message.  
  
  E.g: `json-eval($msg)` |

**Examples**

Following examples use the below sample schema `StockQuoteSchema.json` file. Add this sample schema file (i.e. `StockQuoteSchema.json`) to the following Registry path: `conf:/schema/StockQuoteSchema.json`. For instructions on adding the schema file to the Registry path, see [Adding a Resource](#).

When adding this sample schema file to the Registry, specify the **Media Type** as `application/json`.

---

**Note**

You can also configure the Mediator using XML. Click **switch to source view** in the Mediator window.
Example 1 - Basic configuration

In this example, the required schema for validating messages going through the Validate mediator is given as a registry key (i.e. `schema\StockQuoteSchema.json`). You do not have any source attributes specified. Therefore, the schema will be used to validate the complete JSON body. The mediation logic to follow if the validation fails is defined within the on-fail element. In this example, the `PayloadFactory mediator` creates a fault to be sent back to the party, which sends the message.
<validate>
  <schema key="conf:/schema/StockQuoteSchema.json"/>
  <on-fail>
    <payloadFactory media-type="json">
      <format>{"Error":$1} </format>
      <args>
        <arg evaluator="xml" expression="$ctx:ERROR_MESSAGE"/>
      </args>
    </payloadFactory>
    <property name="HTTP_SC" value="500" scope="axis2"/>
    <respond/>
  </on-fail>
</validate>

An example for a valid JSON payload request is given below.

```json
{
  "getQuote": {
    "request": {
      "symbol": "WSO2"
    }
  }
}
```

**Example 2 - Validate mediator with source (JSONPath)**

In this example, it extracts the message element from the JSON request body and validates only that part of the message against the given schema.

```xml
<validate source="json-eval($.msg)">
  <schema key="conf:/schema/StockQuoteSchema.json"/>
  <on-fail>
    <payloadFactory media-type="json">
      <format>{"Error":$1} </format>
      <args>
        <arg evaluator="xml" expression="$ctx:ERROR_MESSAGE"/>
      </args>
    </payloadFactory>
    <property name="HTTP_SC" value="500" scope="axis2"/>
    <respond/>
  </on-fail>
</validate>
```

An example for a valid JSON request payload is given below.
{
    "msg": {
        "getQuote": {
            "request": {
                "symbol": "WSO2"
            }
        }
    }
}