Orchestration Server Developer's Guide

Orchestration Server Integration

4/18/2025

Contents

Orchestration Server Integration

Orchestrations server integration allows SCXML applications to reach out and interact with other systems within your enterprise and may not only be used for customer specific related integrations but is also leveraged to support integrations to other Genesys products and components that may not as yet have native actions support, such as Context Services and Outbound Contact Server. Such integrations with Genesys components are briefly described in the Genesys Servers section below. Outward integrations from Orchestration are facilitated by the Orchestration Core Extension <fetch> which is defined in more detail in Core Extensions <fetch> which can be consulted to aid in your custom integration.

Genesys Servers

To facilitate easier use of these Genesys-related Servers, Composer supports specific blocks that may remove the need for custom integration code. Because of this, we recommend that you refer to the Composer *Routing Applications User's Guide* for the latest available blocks. This document can be obtained from the Composer page on this wiki.

Context Services

Context Services is a data repository that provides real-time and historic customer-centric data to SCXML applications through an interface that allows the application users to make important business decisions. For example, it allows you to determine whether a customer should be offered a given service or that the customer is a high value client, or if there are existing SCXML application sessions are already running for this customer. Context Services exposes a set of RESTful APIs to access the customer context data. Refer to Context Services for more information about the RESTFul interface and Context Services. The following is an example of how you would use this REST APIs from and SCXML application - details on CS REST API . <!-- This is the IdentifyByPhoneNumber CS API request. -->

```
        <state id="IdentifyByPhoneNumber">
            <onentry>
                <if cond="_data.context_management_services_url == undefined ||
_data.context_management_services_url == ">
                    <raise event="servererror">
                       <param name="description"
expr="'context_management_services_url property not configured'" />
                    </raise>
                <else/>
                    <script>
                        _data.CustomerCount = 0;
                        var includeProfile = "no";
                        var includeExtension = "unique";
                        includeProfile="no";
                        includeExtension="unique";
                    </script>
```

```
                    <session:fetch requestid="_data.requestid"
                                srcexpr="_data.context_management_services_url +
'/profiles'"
                                method="'get'" type="'application/json'">
                            <param name="include_profile" expr="includeProfile"/>
                            <param name="include_extensions"
expr="includeExtension"/>
                            <param name="PhoneNumber" expr="ANI"/>
                    </session:fetch>
                </if>
              </onentry>

            <transition event="error.session.fetch" target="Exit_Error">
                <log expr="'Error ' + _event.data.error + ':' +
_event.data.description" />
            </transition>

            <transition event="session.fetch.done" cond="_event.data.content ==""
target="Exit_Error">
                <assign location="CustomerCount" expr="0" />
                <log expr="'Error No customer found'" />
            </transition>

            <transition event="session.fetch.done" target="Exit_final">
                <script>
                    var _data.CustomerData = _event.data.content !=  ? eval('(' +
_event.data.content + ')') : new Array();
                    if (_data.CustomerData.length == 1) {
                        if (App_IdentifyByPhoneNumber_IncludeExtension=="unique") {
                            _data.CustomerData = [{'customer_id' :
_data.CustomerData[0].customer_id}];
                        }
                    }
                </script>
                <log expr="'IdentifyByPhoneNumber: ' + CustomerData.length + '
Customer record(s) found'"/>
            </transition>

        </state>
```

Outbound Contact Campaigns

The campaign calling list record control actions will be handled through the `<fetch>` action. The Outbound Contact Server campaign-related HTTP APIs will be mapped to the `<fetch>` attributes and child elements. See OCS Support for HTTP Protocol in the *Outbound Contact Reference Manual*.

Important `<fetch>`-related usage notes with the Outbound Web 2.0 APIs:

- The method attribute value is always "post".
- The enctype attribute value is always "application/json".

- The type attribute value is always "application/json".
- The `<param>` name attribute is always "record".

The following is the general mapping to `<fetch>`, while the sub-sections are detailed mappings and examples for the functions that will be supported.

```
<session:fetch requestid="_data.reqid"
               srcexpr="'http://cvserver1.com/<resource>/<id>?req=actionx'"
               method="post" type="application/json" enctype="application/json">
        <param name="record" expr="_data.rmyrecord"/>
</session:fetch>
```

`<resource>` can be:

- records - This contains the campaign records:
  - `<id>` - This will be the ID of the record.
- phones - This is the phone number associated with a record or set of records:
  - `<id>` - This will be the phone number.
- customer_ids - This is the customer ID associated with a record or set of records:
  - `<id>` - This will be the customer ID.

Adding a New Record

This action adds a new record to an existing campaign's call list. This action covers the "Add_Record" IRD function block. Elements:

- `Req = AddRecord`
- `<resource> = records`
- `<id> = recordid`

The following is an example of how to use the `<fetch>` element to add a new outbound record.

```
<onentry>
        <script>
                Record = new Object();
                Record.GSW_PHONE = "567567567545656";
                Record.GSW_TZ_NAME = "'PST";
                Record.GSW_CALL_RESULT = 28;
                Record.STATUS_CODE = "New";
                Record.CUSTOMER_STATUS = 5;
                Record.GSW_CAMPAIGN = "New Productx";
        </script>

        <session:fetch
                        srcexpr="http://server1.genesyslab.com:8080/records/?req=AddRecord"
                        method="post" enctype=" application/json" type=" application/json">
                <param name="record" expr="local.Record"/>
        </session:fetch>
</onentry>
```

Updating an Existing Record

This action updates an existing record in an existing campaign's call list. This action covers the "Update_Record" and "Reschedule" IRD function blocks. Elements:

- `req = RecordReschedule` or `UpdateCallCompletionStats` or `RecordReject` or `RequestRecordCancel`
- `<resource> = records`
- `<id> = recordid`

The following is an example of how to use the `<fetch>` element to reschedule a record.

```
<onentry>
        <script>
                Record = new Object();
                Record.GSW_DATE_TIME = "10/12/2009";
                Record.GSW_CAMPAIGN = "New Productx";
        </script>

        <session:fetch
                        srcexpr="http://server1.genesyslab.com:8080/records/
123456?req=RecordReschedule"
                        method="post" enctype=" application/json" type=" application/json">
                <param name="record" expr="local.Record"/>
        </session:fetch>
</onentry>
```

The following is an example of how to use the `<fetch>` element to UpdateCallCompletionStats for a record.

```
<onentry>
        <script>
                Record = new Object();
                Record.GSW_CAMPAIGN = "New Productx";
        </script>

        <session:fetch
                        srcexpr="http://server1.genesyslab.com:8080/records/
123456?req=UpdateCallCompletionStats"
                        method="post" enctype=" application/json" type=" application/json">
                <param name="record" expr="local.Record"/>
        </session:fetch>
</onentry>
```

The following is an example of how to use the `<fetch>` element to RecordReject a record.

```
<onentry>
        <script>
                Record = new Object();
                Record.GSW_CALLING_LIST = "First List";
                Record.GSW_CAMPAIGN = "New Productx";
        </script>

        <session:fetch
                        srcexpr="http://server1.genesyslab.com:8080/records/
123456?req=RecordReject"
                        method="post" enctype=" application/json" type=" application/json">
                <param name="record" expr="local.Record"/>
        </session:fetch>
```

```
</onentry>
```

Reschedule an Existing Record

This action reschedules an existing record in an existing campaign's call list. This action covers the "Reschedule" IRD function blocks. Elements:

- `req = RecordReschedule`
- `<resource> = records`
- `<id> = recordid`

The following is an example of how to use the `<fetch>` element to reschedule a record.

```
        <onentry>
        <script>
                Record = new Object();
                Record.GSW_DATE_TIME = "10/12/2009";
                Record.GSW_CAMPAIGN = "New Productx";
        </script>
        <session:fetch
                        srcexpr="http://server1.genesyslab.com:8080/records/
123456?req=RecordReschedule"
                        method="post" enctype=" application/json" type=" application/json">
                <param name="record" expr="local.Record"/>
        </session:fetch>
</onentry>
```

Reject an Existing Record

This action rejects an existing record in an existing campaign's call list. Elements:

- `req = RecordReject`
- `<resource> = records`
- `<id> = recordid`

The following is an example of how to use the `<fetch>` element to RecordReject a record.

```
<onentry>
        <script>
                Record = new Object();
                Record.GSW_CALLING_LIST = "First List";
                Record.GSW_CAMPAIGN = "New Productx";
        </script>
        <session:fetch srcexpr="http://server1.genesyslab.com:8080/records/
123456?req=RecordReject"
                        method="post" enctype=" application/json" type=" application/json">
                <param name="record" expr="local.Record"/v
        </session:fetch>
</onentry>
```

Complete an Existing Record

This action completes an existing record in an existing campaign's call list. It covers the "Processed" IRD function block. Elements:

- `req = RecordProcessed`
- `<resource> = records`
- `<id> = recorded`

The following is an example of how to use the `<fetch>` element to RequestRecordCancel a record.

```
<onentry>
        <session:fetch
        srcexpr="http://server1.genesyslab.com:8080/records/123456?req=RequestRecordCancel"
                method="post" enctype=" application/json" type=" application/json"/>
</onentry>
```

Add the Customer to Do Not Contact List

This action completes an existing record in an existing campaign's call list and adds it to the do not contact list. This action covers the "Do Not Call" IRD function block. Elements:

- `<code>req = DoNotCall</code>`
- `<resource> = records`
- `<id> = recordid`

The following is an example of how to use the `<fetch>` element to RecordProcessed a record.

```
<onentry>
        <session:fetch
                srcexpr="http://server1.genesyslab.com:8080/records/123456?req=DoNotcall"
                method="post" enctype=" application/json" type=" application/json"/>
</onentry>
```