Outbound Contact Reference Manual

SCXML Architecture

4/6/2026

SCXML Architecture

Contents

In order to use SCXML-based treatments, Outbound Contact makes use of three components/applications:

- OCS (see Outbound Contact Server)
- Application Server
- SCXML engine/state machine built into OCS

Application Server

SCXML-based treatments are typically hosted/stored on an Application Server (along with other Genesys SCXML scripts, such as SCXML-based strategies) and provided to OCS based on parameters contained in HTTP requests. The `treatment-uri` option identifies the location of these scripts.

Application Server is a web server used to store and retrieve SCXML scripts. You deploy SCXML-based treatments to your production environment by publishing them to an Application Server. Upon an HTTP request, the Application Server is responsible for providing the treatment logic to OCS in the form of a document.

Document Retrieval from Web or Application Server

OCS connects to and retrieves SCXML documents (flexible treatments descriptions) from the Web or Application Server in the following ways:

- OCS uses a connection pool and HTTP/1.1 persistent connections. The size of the pool and connection timeout settings are controlled by the `http-connection-pool-size` and `http-response-timeout` options. Previously, OCS used HTTP 1.0 non-persistent connection to the Web or Application Server.
- OCS opens only a single connection to the given Web or Application Server to retrieve all SCXML documents from that server. Previously, OCS opened as many connections as there were simultaneous requests for SCXML documents from the loaded chains.
- OCS supports secure HTTP (HTTPS) with TLS protocol as a sub layer under regular HTTP for SCXML document retrieval from Web or Application Server. For more details about how to configure and use HTTPS, see Pre-Dial Validation Over Secure Connection.

Supported Application Servers

Genesys supports the following types of Application Server software:

- Microsoft Internet Information Services (IIS), formerly called Microsoft Internet Information Server). Genesys supports IIS 6..19.198.1.
- JBoss Application Server (or JBoss AS). This free software/open source Java EE-based Application Server is usable on any operating system that Java supports. Genesys supports version JBoss 8.1.
- IBM's WebSphere Application Server (WAS). This software Application Server, which is built using open standards (such as Java EE, XML, and Web Services) works with a number of web servers. Genesys supports IBM Websphere Application Server 5.0—8.1.
- Apache/Tomcat

Note:	While Web or Application Server is a more common and a more convenient way to store SCXML documents, you can simply place a file with the SCXML script in your file system. In this case, the absolute path to this file is configured as follows (example): `treatment-uri=file:///C:/GCTI/OCS/Scripts/sample01.scxml`

SCXML Engine/State Machine

The SCXML engine is a subcomponent of OCS that leverages the Genesys SCXML library built into OCS, and which parses the SCXML treatment script. When each record or chain of records associated with the campaign is accessed for processing, the engine creates an instance of the state machine and requests the SCXML treatment from the Application Server. When the record is processed (for example, the call is answered and then completed), the associated state machine instance is stopped.

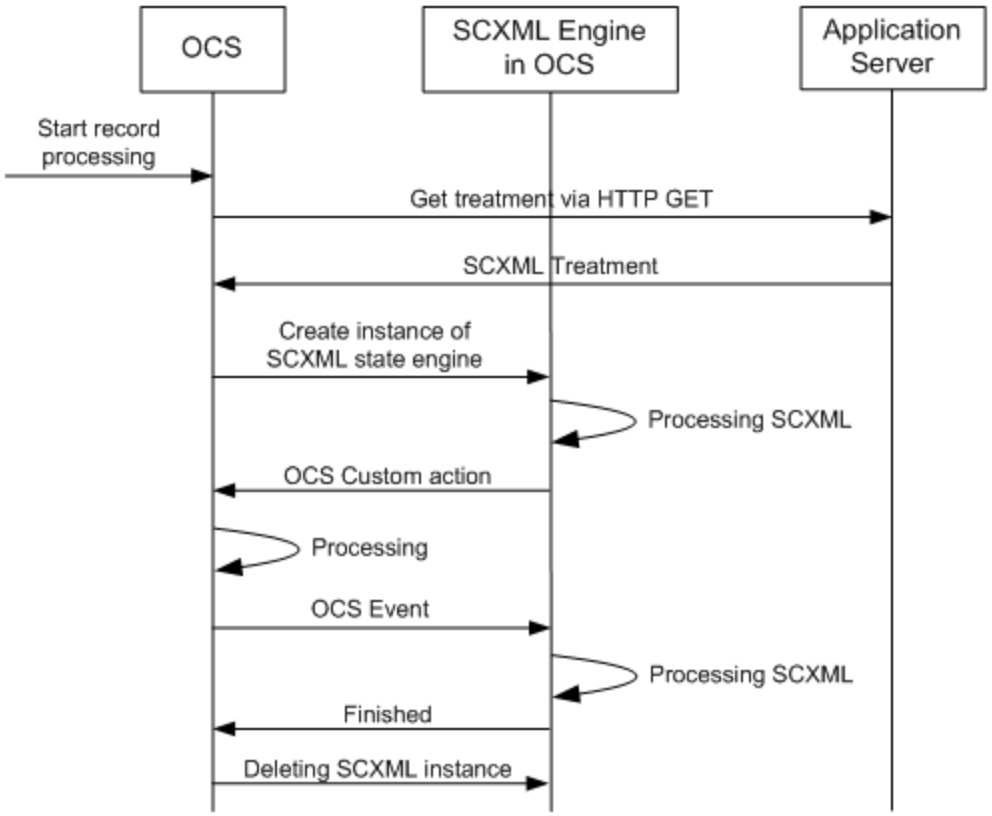
Note:	You specify the SCXML document that describes the treatment used by OCS in the `treatment-uri` option in the Campaign Group or Calling List object. For more information on configuring treatments, see Design and Configuration Task Summary.

A state machine is an SCXML treatment script launched by the SCXML engine, that includes states, actions, events, and transitions, designed to inform OCS on how to contact customers during a campaign.

- *States* provide information on the current status. A state machine operates in one state at a time, until an OCS event provides information that requires a state change, according to the design of the treatment script.
- *Actions* are activities that the state machine performs, according to the treatment script.
- *Events* are messages that OCS sends to the state machine in response to an action.
- *Transitions* occur when the state changes in response to an event.

The following figure illustrates the scenario in which the SCXML engine is used to apply an SCXML treatment.

SCXML Engine/State Machine Treatment Processing describes the diagram.



SCXML Treatment Processing

SCXML Engine/State Machine Treatment Processing

1. OCS receives a request to start processing a record of a Calling List associated with a campaign.
2. As configured in the Campaign Group or Calling List object, OCS requests the associated treatment from Application Server.
3. Application Server provides OCS with the SCXML treatment.
4. OCS requests the Genesys SCXML engine to create and start an instance of the SCXML state machine for the record.
5. The SCXML engine creates and starts the instance and requests that OCS execute customer actions for the record, according to the SCXML treatment script.
6. OCS initiates the requested action (for example, make a call, go to the next record, delay, and so on); for more information on possible actions, see Outbound Contact Custom Actions and Events).
7. After completing each action, OCS sends an event communicating the action result of the action to the instance of the state machine that SCXML engine is running.
8. OCS continues processing the actions, as defined in the state machine.
9. When the processing of the chain of records is completed and chain is finalized, OCS deletes the instance of that state machine.

Note:	A state machine may not be able to complete the

	script for a chain of records if: • The campaign is unloaded. • OCS shuts down.

After a particular instance of the state machine is restored, OCS sends the `ocs.daytime_change` event to the instance so it can adjust its behavior according to the current daytime interval.

Outbound Contact Custom Actions and Events

The following tables describe the custom actions and events supported in Outbound Contact for SCXML treatments.

Custom Actions

Action	Action Details
`timeout`	Response event: `ocs.timeout`
	Element attribute: `delay`--Timeout in seconds
	Child elements: none
	Description: Waits for the specified time.
	Examples • A five minute delay: `<ocs:timeout delay="300"/>` • A delay to the nearest holiday: `<ocs:timeout delay="HDAY"/>`
`next_record`	Reponse event: `ocs.next_record`
	Element attribute: `repeat_chain`-- (Optional) Controls whether the chain should be cycled again starting with the record at the beginning of the chain.
	Child elements: `contact_info_type`0 or more elements can be applied.
	Description: Requests the next record in the chain. The next record is selected according to the criteria defined by the `contact_info_type` child elements (for example, `type_id` or `type_name`). **Note:** An empty `record_id` indicates that there is no next record.
`contact_info_type`	Element attributes: • `type_id`--ID of the contact information type, as specified in the `contact_info_type` field in the Calling List

- type_name--Mnemonic name of the contact information type.
  The list identifies each possible IDs and its corresponding type name:
- 0--NO_TYPE
- 1--HOME
- 2--DIRECT_BUSINESS
- 3--BUSINESS_WITH_EXTENSION
- 4--MOBILE
- 5--VACATION
- 6--PAGER
- 7--MODEM
- 8--VOICE_MAIL
- 9--PIN_PAGER
- 10--E-MAIL

- exclude--The value that determines whether records that match these criteria are excluded from the result. A value of no (the default) includes the records. A value of yes excludes the records.

Child elements: none

Description: Identifies the attribute for contact_type_info. At least one attribute (type_id or type_name) must be specified. If both are specified, type_id has priority.

Note: If both are specified, confirm that the type_id and type_name values match each other. Otherwise, a warning appears in the log.

Examples

Example 1, Getting Next Record with the Home Phone

```
<ocs:next_record>
<ocs:contact_info_type type_id=1 />
</ocs:next_record>
```

Example 2, Getting the Next Record with Phone Other than Home and Mobile

```
<ocs:next_record>
<ocs:contact_info_type type_name="'HOME'"
exclude="'yes'" />
<ocs:contact_info_type type_name="'MOBILE'"
exclude="'yes'" />
</ocs:next_record>
```

Example 3, Getting the Next Record According to the Database Order

```
<ocs:next_record/>
```

	Example 4, Cycling Through the Chain and Getting the Next Record from the Beginning of the Chain <ocs:next_record repeat_chain="'yes'"/>
get_daytime	Request event: daytime_change
	Element attributes: none
	Child elements: none
	Description: Requests information about the current daytime interval.
get_info	Request event: ocs.info

	Element attributes:
	• what--(Mandatory) A string with the type of information being requested. The supported value is TIMERANGE.
	• param--(Optional) A string that is the parameter of the request. For the TIMERANGE type of information, valid values include:
	  • BBH: Before business hours on a weekday
	  • BH: Business hours on a weekday
	  • ABH: After business hours on a weekday
	  • WEND: Weekend
	  • HDAY: Holiday

	Child events: none
	Description: Requests the execution of content-related information from OCS. TIMERANGE, param values = <BBH, BH, ABH, WEND, or HDAY> requests the time period (in seconds) when the specified param attribute timerange begins for the current record.
	Example<ocs:get_info what= "'TIMERANGE'" param = "'HDAY'"/>
make_call	Response event: ocs.callresult
	Element attribute: record_id--ID of the record for which the call will be made.
	Child elements: none
	Description: As a result of this action, OCS initiates a request to make a call using the contact information specified in this record.
	Example:<ocs:make_call record_id="record_id"/>
set_flex_attr	Response event: None if successfulError.Attribute if an invalid value is configured

Element attributes: `record_id`--(Mandatory) Defines the ID of the record (record handle) for which the flexible attribute(s) (options, user data, or Attribute Extensions) are configured. This element can contain a special value of `0`, which is interpreted as `for all records in chain`. For example, it specifies that the same flexible attribute(s) (options) must be set for all the records in the chain that are processed.

Child elements: `flex_attr1` or more elements can be defined.

Child element attributes:

- `attr`-- (Mandatory) Specifies the type of flexible attribute being configured.

Allowed value: `OPTIONS`, `USERDATA`, `EXTENSIONS`

- `type`--(Mandatory) Specifies the type of value that is used for the flexible attribute.

Allowed values: `string` or `int`

- key--(Mandatory) Specifies the mnemonic name of the flexible attribute.

(option name)* `value`--(Mandatory) Specifies the value that is assigned to the attribute. (option value)Element description: Defines a single option, including its name, type, and value.

Descriptions:

`OPTIONS`--This custom action can be used to set one or more options for the specified record in the chain or for all records in the chain. The actual options and values are defined by the `flex_attr` child element(s).
`USERDATA`--Used to update any mandatory or user-defined field that belongs to the current record or chain of records.
`EXTENSIONS`--Used to pass an arbitrary key-value pair in the `AttributeExtensions` field of the Dialing request that OCS sends to the dialer (for example, to T-Server).

- key - specifies the key in the pair
- value - specifies the value for the key

Examples:

1. Set options for the current record:

```
<ocs:set_flex_attr record_id="_event.data.record_id">
<ocs:flex_attr attr="'OPTIONS'" type="'string'"
key="'CPNDigits'" value="'8884554040'"/>
<ocs:flex_attr attr="'OPTIONS'" type="'string'"
key="'pre-dial-validation'" value="'false'"/>
</ocs:set_flex_attr>
```

2. Set option for all records in the chain:

```
<ocs:set_flex_attr record_id="'0'">
<ocs:flex_attr attr="'OPTIONS'" type="'string'"
key="'pre-dial-validation'" value="'true'"/>
<ocs:flex_attr attr="'USERDATA'" type="'string'"
key="'UserField'" value="'UserValue'"/>
<ocs:flex_attr attr="'EXTENSIONS'" type="'string'"
```

	key="'DialAttr'" value="'AttrValue'"/> </ocs:set_flex_attr>
set_exec_flag	Response event: None if successfulError.Attribute if an invalid value is configured
	Element attributes: • flag--(Mandatory) Defines the type of execution flag. Allowed Value: PostProcessing • value--(Mandatory) Defines the value of the execution flag. For PostProcessing flag type, valid values are: Finalize and Switch.
	Child elements: None
	Description: This custom action allows the flag, that controls SCXML treatment execution, to be set. • PostProcessing flag--Determines how the chain of records is processed when the SCXML treatment execution is completed. • Finalize--(default) Processing of the chain is done, chain is finalized (marked completed) in the database. • Switch--Processing of the chain continues after the SCXML treatment execution ceases, and traditional treatments (if configured) take over execution.
	Examples: 1. Switch to traditional treatments: <:set_exec_flag flag="'PostProcessing'" value="'Switch'"/> 2. Finalize chain processing: <:set_exec_flag flag="'PostProcessing'" value="'Finalize'"/> **Note:** This custom action is available in OCS release 8.0.001 and beyond.

Events from OCS

Event	Event Details
ocs.callresult	Response for make_call
	Event properties: callresult--The result of the call. Values include: • Ok • GeneralError

	• SystemError • RemoteRelease • Busy • NoAnswer • SIT_Detected • AnswMachine • All_Trunks_Busy • SIT_Invalid_Num • SIT_Vacant • SIT_Oper_Intercept • SIT_Unknown • SIT_No_Circuit • SIT_Reorder • Fax • Abandoned • Dropped • DroppedNoAnswer • Unknown • Silence • Answer • NuTone • NoDialTone • NoProgress • NoRingBack • NoEstablished • PagerDetected • WrongParty • DialError • CallDropError • SwitchError • NoFreePortError • TransferError • Stale

	• AgentCallBackError • GroupCallBackError • DoNotCall • CancelRecord • WrongNumber
	Description: OCS notifies the state machine instance of the call result.
ocs.daytime_change	Response for get_daytime or an unsolicited request
	Event properties: daytime_code--Code that defines the current daytime interval. Values include: • BBH: Before business hours on a weekday • BH: Business hours on a weekday • ABH: After business hours on a weekday • WEND: Weekend • HDAY: Holiday
	Description: Notifies instances of the SCXML scripts/state machine when the daytime interval changes. If the script is designed to recognize the change, it can result in a different number being dialed (for example, if the daytime interval changes from BH to ABH, a script could be configured to dial the home phone instead of a business phone).
ocs.info	Response for get_info.
	Event properties: result--A string that contains the information requested from OCS.
	Description: Returns information, based on what was requested from OCS in the context of the current record.
ocs.next_record	Response for next_record
	Event properties: • record_id--The OCS-assigned ID of the record in the Calling List. If it is equal to NULL, no next record is available and no other event properties are assigned • contact_info--The value of contact_info field for the record in the Calling List. • <pseudo-fields>--Helper fields defined by OCS including:

	• `gsw_chain_attempts:` (integer) Stores the number of attempts for each chain, which equals the sum of the attempts field values for all records that compose this chain. • `gsw_preferred_flag:` (string) Indicates whether the chain was prioritized by a custom field value, as defined by the `treatment-preferred-contact-field` option. Valid values for this event are `yes` and `no`. • All other fields defined in the Calling List.
	Description: Returns the next record according to the criteria set in the `next_record` custom action.
`ocs.timeout`	Response for `timeout`
	Event properties: none
	Description: Indicates that the `delay` specified in the `timeout` custom action has expired.
`Error.Attribute`	Response for any custom action.
	Event properties: `Error_code`--An error code (integer) that specifies the error that occurred.
	Description: Indicates that an incorrect attribute value is included in the custom action which triggered this event--for example, if the timeout custom action includes a negative delay value or the make_call custom action includes a record_id value that does not exist.
`Error.Configuration`	Response for any custom action
	Event properties: `error_code`--An error code (integer) that specifies the error that occurred.
	Description: Indicates a configuration error--for example, if the timeout custom action includes a delay value of `WEND` but the Statistical Table for weekdays is not configured.