Orchestration Server Developer's Guide

Queue Interface

4/3/2025

Contents

Queue Interface

This Interface provides the ability to allow the orchestration logic to request the appropriate resource for some processing and return the appropriate address information for the resource. The current URS functionality (queuing, prioritization, and so on) will be used for this interface.

Target Formats

All target definitions are strings, but the strings have a particular format, depending on where they are used.

The following is the list of target formats that are used:

- **Target DN:** `<number>@<switch>`.DN. This is the format that is used for a DN that is being used as a target. The following are the details of the different elements of the sub-strings:
  - number is the address of the target resource.
  - switch is the switch or media server that controls this address.
  - DN is the type of target.
- **Possible Target:** `{threshold}[weight] name@server.type` or `{threshold}[weight]?name:skillexpression@server.type`. This is the format that is used when defining a target that should be considered as a possible target or group of targets during the target selection process started by the `<submit>` action. The following are the details of the different elements of the sub-strings.
  - weight - same as defined in `<target>` element
  - name - same as defined in `<target>` element
  - server - same as defined in statserver attribute in the target> element
  - type - same as defined in `<target>` element but with the following abbreviations:
    - A (for agent),
    - AP (for agent place),
    - GA (for group of agents),
    - GP (for group of places),
    - Q (for Queues (real and virtual)),
    - DN (for DN),
    - GA (for skill),
    - C (for campaigns),
    - GC (for group of campaigns),
    - RP (for routing points (real and virtual)),

    - DL (for destination label),
    - GQ (for group of queues (DNs)).
  - skillexpr - same as defined in `<target>` element
  - threshold - same as defined in `<target>` element

Skill Expressions

All skill expression definitions are strings, but the strings have a particular format. A skill expression is a set of conditional expressions that are linked together via logic operators. When a skill expression contains more than one conditional expression, the conditional expression to the left has precedence over the conditional expression to the right. Parentheses can be used to overrule this precedence. Your can use an ECMAScript logic expression to create the skill expression string. The following are the parts of the skill expression:

- **Conditional Expression** - This is an expression that produces a result of true or false. This expression has the following format and elements: **dataname operator value** - for example; english >3
  - **dataname** - This element is a string which represents the name of these data types:
    - **Skill** - This is the name of a skill defined in the configuration layer. Skill names are limited to alphanumeric characters and underscores. The name cannot begin with a digit. The name of a skill cannot exceed 126 characters in length.
    - **Statistic** - This is the name of a statistic defined in the configuration layer. The statistic name in a skill expression can be any agent statistic used in the function SData. It must be written in the format: $(statisticname). For example; $(StatAgentLoggedIn)=1
  - **operator** - This is the operator to be used to evaluate the condition. The following are the operators that are supported:
    - != - The meaning or support is different depending on the data type:
      - Skill - not equal to the indicated level value
      - Statistic - not equal to the indicated statistic value
    - < (<=;) - The meaning or support is different depending on the data type:
      - Skill - less than the indicated level value. **Note:** depending on how you use this operator, it may result in including agents that do not have the skill at all (skillname = 0). For example, with English <= 8, the queue functional module includes all agents with the English skill less than 8, and also agents with no English skill at all.
      - Statistic - less than the indicated statistic value
    - <= (<=) - The meaning or support is different depending on the data type:
      - Skill - less than or equal to the indicated level value
      - Statistic - less than or equal to the indicated statistic value
    - = - The meaning or support is different depending on the data type:
      - Skill - equal to the indicated level value
      - Statistic - equal to the indicated statistic value

        - > (>) - The meaning or support is different depending on the data type:
            - Skill - greater than the indicated level value
            - Statistic - greater than the indicated statistic value
        - >= (>=) - The meaning or support is different depending on the data type:
            - Skill - greater than or equal to the indicated level value
            - Statistic - greater than or equal to the indicated statistic value
    - **value** - This is a value of the same data type as the dataname element. The value must already evaluate to an integer. Floating point numbers are not supported. There are different limitations depending on the data type:
        - **Skill value** - This value represents the level of skill - for example, is the agent's English skill level greater than 3 `(English > 3)`. An agent can be excluded from a skill by setting that agent's skill level for that skill to zero in the configuration layer (that is, English =0).
        - **Statistic** - This value represents the value of the statistic - for example, has the agent been ready longer than 20 seconds`($(StatTimeInReadyState) > 20)`.
- **Logic Operators:** The logic operators are a way to evaluate multiple conditional expressions together. The following logic operators are supported: AND(&- &) and OR (|). The AND and OR logic operators have the same priority. For example; "`English > 3 & $(StatAgentLoggedIn)=1`"

A skill expression has the following limitations:

- The maximum size of an overall skill expression (as text) is 1,024 bytes.

Routing Rules

Currently, Universal Routing Server supports the use of routing rules which are stored in Configuration Server. These are really not rules, but routing profiles which define a unique set of attributes and child elements for a `<submit>` element. An application can use the `<submit>` src attribute for using routing rules. These routing rule definitions can come from transaction objects from Configuration Server.

For details, see <submit>. The following are the properties for each routing rule and how they map to the `<submit>` attributes and child elements:

Routing Rule	Routing Rule Property	Queue Functional Module Interface equivalent
LoadBalancing	VQ name	`<submit>` queue attribute
	List of ACDQueue data (switch and name)	`<targets>` and `<target>` or `<targetid>`
Percentage		`<submit>` ordertype attribute ="percentage"

Routing Rule	Routing Rule Property	Queue Functional Module Interface equivalent
	VQ name	`<submit>` queue attribute
	Type of targets	`<targets>` type attribute
	List of targets and percentages	`<targets>` `<target>` (name and weight attributes)
	Busy treatment data	`<runtreatments>` and all child elements
Service Level		`<submit> ordertype` attribute = "`min`" or "`max`"
	VQ name	`<submit> queue` attribute
	Statserver name	`<targets>` `<target>` `statserver` attribute
	Statistic name	`<submit> orderstat` attribute
	Skills to use and their skill levels	`<target> skillexpr` attribute
	Service Factor data (distribute X percent of interaction in Y seconds)	Not supported through an attribute or child element but can be supported through `<submit> src` attribute and the `gdata` scheme.
	Busy treatment data	`<runtreatments>` and all child elements
	Importance	Not supported through an attribute or child element but can be supported through `<submit> src` attribute and the `gdata` scheme.
Statistics		`<submit>` ordertype attribute = "min" or "max"
	VQ name	`<submit>` queue attribute
	Statserver name	`<targets>` `<target>` statserver attribute
	Statistic name	`<submit>` orderstat attribute

Routing Rule	Routing Rule Property	Queue Functional Module Interface equivalent
	Type of targets	`<targets>` type attribute
	List of target names	`<targets>` `<target>` or `<targetid>`
	Busy treatment data	`<runtreatments>` and all child elements
Workforce	VQ name	`<submit>` queue attribute
	Schedule data (activity name, cut off time)	`<activity>`
	Busy treatment data	`<runtreatments>` and all child elements

Object Model

_genesys.queue Object

This is the global root object for the Queue functional module interface. This object is maintained by the Queue functional module that implements this interface.

The name of the object will be "**_genesys.queue**". There are currently no data properties associated with this object.

_genesys.queue.overwriteType ENUM Object

This represents the DNIS overwrite type enumeration. This enumeration is maintained by the orchestration platform.

This is the set of properties for the object:

Name	Access	Type	Default Value	Valid Values	Description
UseNone	read only	integer	none	0	Use nothing to overwrite the DNIS.
UseANI	read only	integer	none	1	Use the ANI value to overwrite the DNIS.

Name	Access	Type	Default Value	Valid Values	Description
UseDNIS	read only	integer	none	2	Use the DNIS value to overwrite the DNIS.
UseConfig	read only	integer	none	3	Use a configuration value to overwrite the DNIS.
UseValue	read only	integer	none	4	Use the supplied value to overwrite the DNIS.

_genesys.queue.rType ENUM Object

This represents the rType enumeration. This enumeration is maintained by the orchestration platform.

This is the set of properties for the object:

Name	Access	Type	Default Value	Valid Values	Description
RouteTypeUnknown	read only	integer	none	0	This represents an unknown route type
RouteTypeDefault	read only	integer	none	1	This represents a default route type
RouteTypeLabel	read only	integer	none	2	This represents a label route type
RouteTypeOverwriteDNIS	read only	integer	none	3	This represents an overwrite DNIS route type
RouteTypeDDD	read only	integer	none	4	This represents a DDD route type
RouteTypeIDDD	read only	integer	none	5	This represents an IDDD route type
RouteTypeDirect	read only	integer	none	6	This represents a direct route type
RouteTypeReject	read only	integer	none	7	This represents a

Name	Access	Type	Default Value	Valid Values	Description
					reject route type
RouteTypeAnnouncement	read only	integer	none	8	This represents an announcement route type
RouteTypePostFeature	read only	integer	none	9	This represents a post feature route type
RouteTypeDirectAgent	read only	integer	none	10	This represents a direct agent route type
RouteTypePriority	read only	integer	none	11	This represents a priority route type
RouteTypeDirectPriority	read only	integer	none	12	This represents a direct priority route type
RouteTypeGetFromDN	read only	integer	none	13	This represents a from DN route type
RouteTypeAgentID	read only	integer	none	14	This represents an agent ID route type
RouteTypeCallDisconnect	read only	integer	none	15	This represents a call disconnect route type

_genesys.queue.quotaType ENUM Object

This represents the quotaType enumeration. This enumeration is maintained by the orchestration platform.

This is the set of properties for the object:

Name	Access	Type	Default Value	Valid Values	Description
QuotaMin	read only	integer	none	0	This means to filter on the minimum quota values
QuotaTarget	read only	integer	none	1	This represents

Name	Access	Type	Default Value	Valid Values	Description
					means to filter on the target-based quota values
QuotaMax	read only	integer	none	2	This means to filter on the maximum quota values

_genesys.queue.statcond ENUM Object

This represents the statcond enumeration. This enumeration is maintained by the orchestration platform.

This is the set of properties for the object:

Name	Access	Type	Default Value	Valid Values	Description
ReadyIfLess	read only	integer	none	0	This represents the condition where the agent is ready and the associated statistic is less than the threshold value.
ReadyIfGreater	read only	integer	none	1	This represents the condition where the agent is ready and the associated statistic is greater than the threshold value.
ReadyIfNotGreater	read only	integer	none	2	This represents the condition where the agent is ready and the associated statistic is not greater than the threshold value.
ReadyIfNotLess	read only	integer	none	3	This represents the condition where the agent is ready and the associated statistic is not less than the threshold value.

_genesys.queue.usecapcond ENUM Object

This represents the usecapcond enumeration. This enumeration is maintained by the orchestration platform.

This is the set of properties for the object:

Name	Access	Type	Default Value	Valid Values	Description
OnStatError	read only	integer	none	0	This indicates to use the statistical tables from the configuration layer when there is an error with the Stat Server results.
Never	read only	integer	none	1	This indicates to never use the statistical tables
Only	read only	integer	none	2	This indicates to only use the statistical tables.

Functions

_genesys.queue.reserveTarget

When the `<submit>` action returns a ready target, the target is always blocked by the functional module during some time in seconds (the value of the platform's *transition_time* configuration option). The reserveTarget function allows you to override such behavior and change the time interval or even cancel it entirely (*time* parameter = 0). During the time the target is reserved, the functional module does not distribute any interactions to the target as specified by the target parameter, which is the result of the `<submit>` action. This function can be used in cases where a target may have additional conditions (except a Stat Server-reported not-ready state) that should prevent the target from being selected as a valid target. This blocking time will be applied to all `<submit>` action resulting targets until it is changed again by this function.

`void _genesys.queue.reserveTarget(ixnid, target, time)` Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **target:** STRING which can be a variable or a constant - This is the name of the target to be reserved in the target DN format.
- **time:** INTEGER which can be a variable or a constant - This is the time interval in seconds to reserve this target for (exclude from target selection process). A value of zero (0) will cancel it entirely.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.reserveTarget( _data.ixnid, 'return:ok|n:_data.reqid|agent:703_sip|place:703|
dn:703|switch:SipSwitch', 40 );
```

_genesys.queue.cCTExtractTargets

This function produces a list of targets in "possible target" format from the supplied arguments and stores target DN information associated to each target for subsequent use in number translation of type`[TARGET.CCTN].`These are maintained for the life of the session by the functional module and can be used for any interaction associated with session.

This function supports these target types:

- Agent (A)
- Place (P)
- Agent Group (GA)
- Place Group (GP)

`targets _genesys.queue.cCTExtractTargets(ixnid, statserver, input)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **statserver:** STRING which can be a variable or a constant - This parameter is used as the location attribute for each of the targets listed in the output.
- **input:** STRING which can be a variable or a constant - This parameter must have the following key/value list of the form: TargetID1.TargetType1:Value1|...|TargetIDN.TargetTypeN:ValueN with no spaces in the list.

Returns: `targets`: STRING-The result of the function is a string of the form: `"TargetID1@StatServer.TargetType1,...,TargetIDN@StatServer.TargetTypeN"`. Thus, the targets listed in the result are separated by commas and contain no spaces.

The following is an example of input and output for this function:

```
_data.myResult = _genesys.queue.cCTExtractTargets( _data.ixnid, 'StatServer', '701.A|702.P|
SipGr_3.GA|SipPlGr2.GP' );
```

_data.myResult will be the following string:

```
'701@StatServer.A,702@StatServer.P,SipGr_3@StatServer.GA,SipPlGr2@StatServer.GP'
```

_genesys.queue.checkAgentState

This function instructs the functional module for the associated session (across all associated interactions) as to whether to take into account the state of an Agent, Place, Agent Group, or Place Group as reported by Stat Server or to look only for free DNs belonging to the Agent, Place, or Agent

Group. For example, `_genesys.queue.checkAgentState(false)` makes it possible to route a voice interaction to an agent that Stat Server reports as not ready. If agent capacity rules are set, this function has no effect (as the Genesys Agent Capacity model does not use agent state).

To allow an agent to receive multiple voice interactions, this function must be `false`. However, the side effect is that the functional module distributes interactions so as to occupy all DNs of Agent A before considering Agent B. When this function is set to `false`, or the function `_genesys.queue.useAgentState` is used, the functional module does not apply the *verification_time* configuration option to agents, but still applies it to agent DNs.

`void _genesys.queue.checkAgentState(ixnid, check)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **check:** BOOLEAN which can be a variable or a constant - This parameter identifies whether the functional module should use the agent state from StatServer or not.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.checkAgentState( _data.ixnid, false);
```

_genesys.queue.clearThresholds

This function invalidates all thresholds previously set by `<submit>` actions for a given interaction. As a result, the functional module now considers all targets that were previously affected by `<submit>` actions as unconditionally ready for routing.

`void _genesys.queue.clearThresholds(ixnid)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.clearThresholds( _data.ixnid );
```

_genesys.queue.countSkillInGroup

This function determines the number of agents with a skill set or statistical parameters that satisfy the indicated skill expression. It returns the number of the agents belonging to the agent group based on the defined skill expression. You can also just specify a Stat Server and a skill expression without specifying an Agent Group. However, a skill expression and a Stat Server must be specified. The Stat Server is used to query the content of the provided Agent Group (real or virtual).

`total _genesys.queue.countSkillInGroup(ixnid, statserver, group, sexpr)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **statserver:** STRING which can be a variable or a constant - This parameter is optional. This parameter is the name of the Stat Server containing information on the agents for this function. If not supplied, the Stat Server used will be the one defined by the default_stat_server configuration option of the platform.
- **group:** STRING which can be a variable or a constant - This parameter is the agent group for the Stat Server that this function checks against. It can either be an Agent Group name or a list of comma-separated list of Agents, Agent Groups, Places, or Place Groups. Agents are included in this group either by placing the agent name from the Persons folder into the Agent Groups folder or defining a virtual group using a skill expression within the Annex tab of the Agent Group object.
- **sexpr:** STRING which can be a variable or a constant - This parameter defines the skill expression string used to evaluate the agents. It can use skills, variables, numeric constants, and statistics to filter out agents based on their state. The statistic name in a skill expression can be any agent statistic used in the function SData. It must be written in the format:`$(statistic)`. This ability to use statistics in a skill expression allows you to conduct queries based on a statistic. For example, if you want to query the number of agents with a Spanish skill of at least 5 who are logged in, the expression would be as follows: `'Spanish >= 5 & $(StatAgentsLoggedIn)=1'.` This can be passed to the function as a literal string or as an an ECMAScript expression such as `'Spanish >=' + level + ' $(StatAgentsLoggedIn)=' + minAgents` where level and miNAgents are variables defined else where in your applciation.

Returns: `total`: NUMBER - The result of the function is an integer which represents the number of agents that meet the critieria of the skill expression.

The following is an example of input and output for this function:

```
_data.myResult = _genesys.queue.countSkillInGroup( _data.ixnid, 'StatServer', 'SipGr_2',
'english > 3 & french > 4' );
```

_data.myResult will be the following:

```
5
```

_genesys.queue.createSkillGroup

This function converts the provided Agent Group, Skill Expression, and StatServer into a normal target format that represents all agents belonging to the Agent Group supplied as a parameter that satisfies the logical condition given by the Skill Expression.

`tlist _genesys.queue.createSkillGroup(ixnid, statserver, group, sexpr)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **statserver:** STRING which can be a variable or a constant - This parameter is optional. It contains the name of the Stat Server containing information on the agents for this function. If not supplied, the Stat Server used will be the one defined by the *default_stat_server* configuration option of the platform.
- **group:** STRING which can be a variable or a constant - This parameter is the agent group for the Stat

Server that this function checks against.

- **sexpr:** STRING which can be a variable or a constant - This parameter defines the skill expression used to evaluate the agents. This parameter can use skills, variables, numeric constants, and statistics to filter out agents based on their state. The statistic name in a skill expression can be any agent statistic used in the function SData. It must be written in the format: `$(statistic)`. This ability to use statistics in a skill expression allows you to conduct queries based on a statistic. For example, if you want to query the number of agents with a Spanish skill of at least 5 who are logged in, the expression would be as follows: `Spanish >= 5 & $(StatAgentsLoggedIn)=1`.

Returns: `tlist`: STRING - The result of the function is a string which represents a list of targets (each target is comma-separated in the string) that meet the critieria of the skill expression. It is of the form:`"?GroupName:SkillExpression@statserver.GA".`

The following is an example of input and output for this function:

```
_data.myResult = _genesys.queue.createSkillGroup( _data.ixnid, 'StatServer', 'SipGr_2',
'english>3' );
```

_data.myResult will be the following string:

```
'?SipGr_2:english>3@StatServer.GA'
```

_genesys.queue.excludeAgents

This function instructs the functional module not to route interactions to any agent on the specified list of agents. This applies across all `<submit>` actions associated with the session.

`prev_agents _genesys.queue.excludeAgents(ixnid, agents)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **agents:** STRING which can be a variable or a constant - This parameter is a comma-separated list of agent IDs.

Returns: `prev_agents`: STRING - The result of the function is a string which represents a list of previous excluded agents. Each agent ID is comma-separated in the string.

The following is an example of input and output for this function:

```
_data.myResult1 = _genesys.queue.excludeAgents( _data.ixnid, '702_sip' );
...
_data.myResult2 = _genesys.queue.excludeAgents( _data.ixnid, '703_sip' );
```

_data.myResult2 will be the following string:

```
'"702_sip"'
```

_genesys.queue.extRouterError

This function is used to change the default external routing in the case of a failure to get a remote access number. This is only applicable across all interactions associated with the session when the application is using the `<submit>` action with the route attribute set to true.

`void _genesys.queue.extRouterError(ixnid, enable)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **enable:** BOOLEAN which can be a variable or a constant - If this parameter is set to true, then the functional module handles an external routing failure as a routing error (according to the functional module on_route_error configuration option settings. This prevents the functional module from ignoring any error messages in response to external routing requests and stops the functional module from continuing to attempt to route based on the original remote access number. If set to false, the functional module will continue the attempt to route the call based on the original number. By default, enable is set to false.

Returns: `void`

The following is an example of input and output for this function:

```
_genesys.queue.extRouterError( _data.ixnid, true );
```

_genesys.queue.extRouteStatus

This function returns true if external routing is possible and false if not. `status _genesys.queue.extRouteStatus(ixnid, switch)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **switch:** STRING which can be a variable or a constant - This parameter is a remote location where the interaction is being routed (that is, the switch ID).

Returns: `status`: BOOLEAN - The result of the function is a boolean which indicates whether external routing is possible for the given media server (switch).

The following is an example of input and output for this function:

```
_data.myResult = _genesys.queue.extRouteStatus( _data.ixnid, 'SipSwitch' );
```

_data.myResult will be the following string:

```
true
```

_genesys.queue.findServiceObjective

This function returns a value that is defined in a configuration layer objective table object for a given a Service Objective, which is a unique combination of Customer Segment, Service Type, and Media Type. If the *Update* parameter is true, the Service Objective with Service Type and Customer Segment are automatically attached to the interaction.

`value _genesys.queue.findServiceObjective(ixnid, table, media, service, segment,` `update)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **table:** STRING which can be a variable or a constant - This parameter is the name of the objective table in the configuration.
- **media:** NUMBER which can be a variable or a constant - This parameter is the desired media type in integer form.
- **service:** STRING which can be a variable or a constant - This parameter defines the type of service desired.
- **segment:** STRING which can be a variable or a constant - This parameter defines the type of customer segment desired.
- **update:** BOOLEAN which can be a variable or a constant - This parameter defines whether the interaction should be updated with a value as well as the properties of the service objective (media type, service type, and customer segment). The update parameter, when set to "true", will work for voice interactions only. For eService interactions, the developer should ensure that the user data is updated, via the Strategy, with the appropriate KVPs (names and values) on receipt of the service objective value.

Returns: `value`: NUMBER - The result of the function is an integer which represents the configured value for this service objective. Zero is returned if there are any issues (unsuccessful search - no service objective defined) while processing this function.

The following is an example of input and output for this function:

```
_data.myResult = _genesys.queue.findServiceObjective( _data.ixnid, 'OBT3', 0, 'help',
'Bronze', true );
```

_data.myResult will be the following number:

```
150
```

_genesys.queue.incrementPriority

This function results in incrementing the selected interaction priority by the `increment` every `interval` second. It affects the priority of the interaction for `<submit>` actions the interaction is already waiting for and those it may be waiting for in the future.

`void _genesys.queue.incrementPriority(ixnid, increment, interval)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have its priority incremented.
- **increment:** NUMBER (INTEGER) which can be a variable or a constant - This parameter is the increment by which the priority is to be adjusted.
- **interval:** NUMBER (INTEGER) which can be a variable or a constant - This parameter is the time interval at which the priority is to be incremented. **Note:** the interval cannot be less than 5 seconds.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.incrementPriority( _data.ixnid, 2, 10 );
```

_genesys.queue.priorityLimits (since ORS 8.1.200.40 and URS 8.1.200.21)

This function results in imposing additional limits on values of priority the interaction can have in any routing queue. Without it interaction priority can accept any value in range [-1,000,000,000 : 1,000,000,000].

`void _genesys.queue.priorityLimits(ixnid, min, max)`

Parameters (all parameters are mandatory):

- **ixnid:** STRING which can be a variable or a constant - This parameter defines the ID of the interaction to which provided priority limits will be applied.
- **min:** NUMBER (INTEGER) which can be a variable or a constant - This parameter is the minimal possible value for the interaction priority in any routing queue.
- **max:** NUMBER (INTEGER) which can be a variable or a constant - This parameter is the maximal possible value for the interaction priority in any routing queue.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.priorityLimits( _data.ixnid, 1, 10 );
```

_genesys.queue.nMTExtractTargets

This function enables you to track the number of active interactions at a device that is not configured in the Configuration Database. It establishes a counter for all active interactions at the non-configured device. Using this counter, you can compare the number of active interactions to a specified threshold and stop routing interactions to the device when the threshold has been reached.

This function parses a list of targets, which is a comma-separated list of attributes produced by a database query or some outside application (an application on an application server, for example) such as `<fetch>`. Each target can be described using four attributes: *Target Name, Threshold, Speaking Time*, and *RetryTime*. These attributes are explained below:

- **Target Name** - This identifies the target.
- **Threshold** - The maximum number of interactions at the target when this target is considered to be ready. The threshold value is considered as attribute to an interaction. This function sets this attribute for each interaction it evaluates. When it is time for the interaction to be routed to the non-configured device, the functional module compares the interaction's threshold attribute to the counter. If the counter shows a value greater than the interaction's threshold attribute, the interaction is not routed until the counter's value drops below that threshold.
- **Speaking Time** - The functional module considers an interaction to a non-configured device to be terminated when the controlling server (for example, T-Server) reports that it has been terminated. However, if no information is received from the controlling server (for example, T-Server) in the time

specified by the Speaking Time (in seconds), the functional module considers the call to be terminated.

- **Retry Time** - If the functional module receives an error message in response to a routing request to a non-configured device that indicates that the number of interactions at the target is different from the number the functional module believes are there, the functional module temporarily stops sending interactions to that device for the amount of time specified by this element (Retry Time). This delay enables synchronization of the number of interactions on the device with the number that the functional module believes are there.

The following is an example of input and output for this function:

```
_data.myResult = _genesys.queue.nMTExtractTargets( _data.ixnid, '701,702,703', 1, 2, 5, 3 );
```

_data.myResult will be the following string:

```
'{RStatCallsInQueue<=2}701.DN,{RStatCallsInQueue<=2}702.DN,{RStatCallsInQueue<=2}703.DN'
```

Using the Function

Use of this function to track the number of interactions at non-configured devices usually includes:

- Setting a threshold, using the `_genesys.queue.setThresholdEx` function, for every nonconfigured device that you want to include. The only threshold you can set for non-configured devices applies to the value returned by the `RStatCallsInQueue` statistic. Alternatively, you can set a threshold by prefixing `Threshold{Statistic op value}` before the target specification. When you set a threshold in this way, this function automatically augments the targets with prefixes in the indicated format. The output of this function is a comma-separated list of targets, with thresholds, that is ready to be used as a parameter of standard target-selecting functions or objects.
- Configuring the functional module to count the `RStatCallsInQueue` statistic for nonconfigured devices.
- Setting all the non-configured devices as targets in `<submit>` actions.

Note: The functional module has no information about the current state of the device - only the number of interactions active on it. Active interactions are considered to be the number of interactions the functional module has sent to the target, minus the number of interactions that were terminated. In addition to generating output, this function also overwrites the default values (15 seconds and 600 seconds) for *Retry Time* and *Speaking Time*. Thresholds for non-configured devices set using the `_genesys.FMname.SetThresholdEx` function, for example, use the current default values for *Retry Time* and *Speaking Time* that were set by the latest invocation of this function.

Note: Changes applied to default settings (*RetryTime, Speaking Time*) specified in this function work only for non-configured devices that were created after this function was used. Those non-configured devices that were created previously are not affected by any changes made using this function. If the information about the non-configured devices (numbers, threshold values) is retrieved from an outside source (for example, a database), use this function to transform the result of this query into a target list suitable for use by the `<submit>` action. In this case, the use of this function automatically sets all the specified thresholds and you do not need to call the `_genesys.FMname.setThresholdEx` function.

Setting Multiple Thresholds

You can set more than one threshold by using this function in more than one logic definition or you can use it multiple times in the same logic. In this way, you can set a different allowable number of concurrent interactions for different situations, as described below.

Note: If you use the same non-configured device in other logic definitions or in multiple places in the same logic, the functional module uses the same counter for that device. So this definition is global across the functional module.

Using `NMTExtractTargets` to Set Multiple Thresholds - Example:

You can set different thresholds for the same non-configured device to take account of different business conditions. For example, between 9:00 AM and 5:00 PM, an outsourcer with a nonconfigured device can handle 100 concurrent interactions. After 5:00PM, the outsourcer can handle only 50 concurrent interactions on this device.

Here is how it works:

1. Place this function in two places. You can use it twice in one logic definition or use two separate logic definitions to handle this situation, depending on what is best suited to your environment. A global counter is created for the non-configured device.
2. Using one instance of this function, set a default threshold value of 100 to be attached as an attribute to all interactions that arrive between 9:00 AM and 5:00 PM.
3. Using the other instance of the function, set a default threshold value of 50 to be attached as an attribute to all interactions that arrive between 5:00 PM and 9:00 AM.

When an interaction is ready to be routed, the interaction attribute is compared to the global counter, which is incremented or decremented to correspond to the number of active interactions on this non-configured device. If the value on the counter is greater than the interaction's Threshold attribute, the interaction is not routed until the counter's value drops below the appropriate threshold.

Deployment Considerations

Only one orchestration platform can send interactions to the non-monitored device. If you require multiple numbers to distribute to the same non-monitored device, align the SN-to-DN table, the network switch, and platform so that all interactions sent to a nonmonitored device are processed by the same platform. You can use a network-controlling server (for example, T-Server or a SIP Server) as the network switch. To use this function, the platform and functional module must register on a special device (called switch::) in order to receive notification from the controlling server (for example, T-Server) about the termination of interactions. If you do not register, the *`RStatCallsInQueue`*statistic is not correctly calculated for non-monitored targets. Registration is not automatic and is controlled by platform (using the *call_monitoring* configuration option).

```
tlist _genesys.queue.nMTExtractTargets(ixnid, targets, dThreshold, dSpeakTime,
dRetryTime, recordSize)
```

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory It defines the ID of the interaction which should have its priority incremented.
- **targets:** STRING which can be a variable or a constant - This parameter is the list of comma-separated targets and their attributes. The recordSize parameter will identify how attributes are associated with a target definition.
- **dThreshold:** NUMBER (INTEGER) which can be a variable or a constant - This parameter is the default maximum number of interactions at the target when this target is considered to be ready. This parameter will be used for targets that do not have this attribute specified in the targets parameter.

- **dSpeakTime:** NUMBER (INTEGER) which can be a variable or a constant - This parameter is the default maximum amount of speak time for an interaction. If the functional module does not receive an event indicating that the interaction has terminated with in this time frame, the functional module will consider it terminated. This parameter will be used for targets that do not have this attribute specified in the targets parameter.
- **dRetryTime:** NUMBER (INTEGER) which can be a variable or a constant - This parameter is the default maximum amount of retry time for an interaction. If the functional module failed to route the interaction, it will wait for this amount of time before trying to route the interaction again. This parameter will be used for targets that do not have this attribute specified in the targets parameter.
- **recordSize:** NUMBER (INTEGER) which can be a variable or a constant - This parameter defines how many attributes are entered to describe each target. If some of the four elements listed in the following bullets [I don't see the bullets...] are absent (only Target name is mandatory) the corresponding default parameter values are used instead.

Returns: `tlist`: STRING - The result of the function is a string of the form: "{RStatCallsInQueue<=TargetThres1}TargetName1.DN,...,{RStatCallsInQueue<=TargetThresN}TargetNameN.DN". Thus, the targets listed in the result are separated by commas and contain no spaces.

_genesys.queue.onRouteError

This function allows you to specify an individual functional module reaction for every type of error. If used, this option overwrites the*on_route_error* configuration option for all interactions associated with the current session. In the case of an error, the functional module behaves in the following way:

- Checks if this function was executed for this type of error. If yes, then the functional module behaves as the function specifies.
- Otherwise, checks if there is a platform-defined default reaction (that is, the configuration option *on_route_error*) for this error. If yes, then the functional module executes this reaction.

Note: This function should ONLY be used when using the `<submit>` action with the route attribute set to true. `void _genesys.queue.onRouteError(ixnid, type, option)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory It defines the ID of the interaction which should have its priority incremented.
- **type:** NUMBER (INTEGER) which can be a variable or a constant - This parameter indicates the type of error that can be encountered while trying to route an interaction.
- **option:** STRING which can be a variable or a constant - This parameter indicates the type of processing the functional module should perform when this type of error is encountered. These options only work for interactions routed to destinations specified in the `<submit>` actions with the route attribute set to true. The values are defined as follows:
  - **try_other** - The Queue functional module will resume waiting for a ready target and try to select another available target. Unlike the reroute value, for which the Queue functional module immediately attempts to reroute the interaction, for try_other the Queue functional module waits for another target, if none are available before routing the interaction. **Note:** Be sure that the *transition_time* platform configuration option has a value of 3 or higher to enable the Queue functional module to handle the *try_other* setting correctly.
  - **strategy_error** - The Queue functional module will abort its current waiting for a ready target, go to error handling, and send the session an error event (`error.queue.submit` with the appropriate error code).

Returns: `void`

The following is an example of input and output for this function:

```
_genesys.queue.onRouteError( _data.ixnid, 140, 'strategy_error' );
```

_genesys.queue.priorityTuning

The functional module always puts interactions into waiting queues according to their priorities. This function defines how the functional module handles interactions with the same priorities. By default, interactions with the same priority are ordered according to the time the interaction began to wait for some target. This applies across all `<submit>` actions.

`void _genesys.queue.priorityTuning(ixnid, useAge,usePredict, useObjective)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory It defines the ID of the interaction which should have its priority incremented.
- **useAge:** BOOLEAN which can be a variable or a constant - This parameter defines whether the age of the interaction should be used to prioritize interactions with the same priority. If this parameter is true, the functional module uses the time the interaction was created instead of the time the interaction is placed into the waiting queue. The age of the interaction is usually the time that the associated session is started for the interaction. Setting the interaction age enables you to safely use multiple `<submit>` actions. The interaction does not lose its position in the queue, because its position is based on the age-of-interaction value, which is not affected by the `<submit>` actions. The following must be done in your logic to use this function.
    - Use function `_genesys.queue.setInteractionAge()` to timestamp the interaction. The age of interaction will be counted from this moment on.
    - Use the function to prioritize this interaction among others in the queue according to their age.
- **usePredict:** BOOLEAN which can be a variable or a constant - This parameter defines if the estimated time for the interaction to be answered will be used instead of the time the interaction has waited when prioritizing interactions with the same priority. If this parameter is true, then the functional module calculates the estimated time for the interaction to be answered and will use this time instead of the time that the interaction has already waited.
- **useObjective:** BOOLEAN which can be a variable or a constant - This parameter defines whether an objective defined in the interaction should be used when prioritizing interactions with the same priority. If this parameter is true, then the functional module calculates the objective defined in the interaction to determine the priority of the interaction.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.priorityTuning( _data.ixnid, true, true, true );
```

_genesys.queue.resetAdjustment

This function cancels any adjustment that may have been set for a statistic for a given target using `_genesys.queue.setAdjustment`.

`void _genesys.queue.resetAdjustment(_data.ixnid, target, statistic)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **target:** STRING which can be a variable or a constant - This parameter defines the target whose statistic adjustment is to be cleared. The string value of this parameter must be in the possible target format. See the Target Formats section for details.
- **statistic:** STRING which can be a variable or a constant - This parameter defines the name of the statistic for which the adjustment is to be removed.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.resetAdjustment(_data.ixnid,'SipGr_2.GA','StatAgentsAvailable');
```

_genesys.queue.routed

This function marks the interaction as routed. When routing functions, such as `<submit>` actions with the route attribute set to true, are successfully executed, an interaction is implicitly marked as routed. This function explicitly marks the interaction as routed. This function allows the functional module to dispose of the interaction when the strategy is completed and not route the interaction to the default destination.

`void _genesys.queue.routed(ixnid)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which routing has terminated.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.routed( _data.ixnid );
```

_genesys.queue.selectTargets

This function removes from a list of Agent Groups, Place Groups, or Queue Targets those targets that have already received a number of calls in excess of their quota for the interval when the function is called. These types of quotas are specified in the configuration layer in the *Statistical Days* belonging to *Statistical Tables* of type *Quota Table*, associated with each of the Agent Groups or Place Groups. The queue is associated with that *Quota Table* that is associated with an Agent Group for which the queue is an origination DN. If no agent group is found, the queue is associated with the *Quota Table* that is associated to an Agent Group named after the alias of the queue. If an Agent Group or Place Group target on the list has no *Quota Table* associated with it or no *Statistical Day* in the table matches the current date, the target is retained in the list returned by the function: it has not exceeded its quota since no quota was set for it.

Configuring Quota Tables Associated to Groups of Agents or Places

A *Quota Table* is configured in the configuration layer as a *Statistical Table* object of *Quota Table* type (see the configuration layer documentation for more details on the process of configuration). The*Quota Table* associated with an Agent Group or a Place Group must be specified inside the Advanced properties of the group. The same*Quota Table* can be associated with more than one Agent Group or Place Group. The *Quota Table* must contain *Statistical Days*. Use the information about *Statistical Days*, but not the information about Statistical Values. The relevant values for Statistical Values are as follows: *Statistical Value 1, Statistical Value 2*, and *Statistical Value 3* for each Interval of the day - during every interval. Value 1 is used when Quota type has a value of 0 (*QuotaMin*). Value 2 is used when Quota type has a value of 1 (*QuotaTarget*). Value 3 is used when Quota type has a value of 2 (*QuotaMax*). All other properties of Statistical Days are irrelevant for the purpose of setting up quotas. **Note:** The same *Statistical Day* can belong to more than one *Quota Table*.

`tlist _genesys.queue.selectTargets(ixnid, filter, targets)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **filter:** genesys.queue.quotaType ENUM OBJECT which can be a variable or a constant - This parameter indicates which of the three relevant entries in the statistical day will be treated as the current quota.
- **targets:** STRING which can be a variable or a constant - This parameter is a string of comma-separated high-level Agent Groups Or Place Groups or Queue Targets (in the possible target format, see the Target Formats section for details). If this parameter has queue targets, you need to make sure that every queue for which you use this parameter is listed as an origination DN for at most one Agent Group in the configuration layer. If more than one group or queue is associated with the same table, then the interactions routed to all of them are counted together. That is, the quota is interpreted as a limit on the total number of interactions routed to groups and queues associated with the same table. Therefore, you must set up individual Quota Tables for groups and queues that you want to consider separately, even if these tables consist of the same Statistical Days.

Returns: `tlist`: STRING - The result of the function is a string of targets in the possible Target Formats that matches the quota filter criteria. However, if the function encounters a list element in a different format, it does one of the following:

- If the element is a syntactically correct target in the complete high-level format but its type is not Group of Agents or Group of Places, the target is retained in the list returned by the function.
- If the element is not a syntactically correct target in the complete high-level format, including targets with an omitted type or location, it will be dropped from the list.

The following is an example of input and output for this function:

```
_data.myResult = _genesys.queue.selectTargets( _data.ixnid, 1,
'SipGr_3@StatServer.GA,SipPlGr2@StatServer.GP' );
```

_data.myResult will be the following string:

```
'SipGr_3@StatServer.GA,SipPlGr2@StatServer.GP'
```

_genesys.queue.selectTargetsByThreshold

This function finds the best available target(s) from a list of targets by applying a statistic with a threshold comparison against the input target list. This function returns a subset of the `targets` List parameter (possibly empty).

`tlist _genesys.queue.selectTargetsByThreshold(ixnid, targets, statistic, value, cond)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **targets:** STRING which can be a variable or a constant - This parameter is a string of comma-separated high-level targets (in the possible target format, see the Target Formats section for details).
- **statistic:** STRING which can be a variable or a constant - This parameter is a statistic to be used in the comparison.
- **value:** NUMBER (INTEGER) which can be a variable or a constant - This parameter is the value that will be used in the comparison to see if a target meets the condition or not.
- **cond:** genesys.queue.statcond ENUM OBJECT which can be a variable or a constant - This parameter is the condition which is to be used in the comparison.

Returns: `tlist`: STRING - The result of the function is a string of targets in the possible Target Formats that matches the threshold filter criteria.

The following is an example of input and output for this function:

```
_data.myResult = _genesys.queue.selectTargetsByThreshold( _data.ixnid,
'701_sip.A,702_sip.A,703_sip.A', 'StatCallsAnswered', 3, 0 );
```

_data.myResult will be the following string:

```
'702_sip@StatServer.A,703_sip@StatServer.A'
```

_genesys.queue.setInteractionAge

This function overrides the default age of an interaction from a routing perspective. This function can be useful if the age of the interaction will be used for placing interactions into waiting queues. The interaction age is the time accumulated since the interaction was known by Genesys (normally set at the very first resource or device the interaction enters).

By default, the interaction age is defined by the time of the last event (for example, `EventRouteRequest`). However, if an interaction is going to be routed more than once (for example, if an agent transfers an interaction on a routing point for re-routing or if the Voice Callback Universal Callback Server resubmits a callback interaction to URS), the time of the last event (for example, `EventRouteRequest`) is not always the best way to define interaction age.

`void _genesys.queue.setInteractionAge(ixnid, keep)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **keep:** BOOLEAN which can be a variable or a constant - This parameter indicates whether the default age algorithm should be. If this parameter is true, it fixes the current interaction age so that it does not depend on subsequent routing events. If the parameter is false, it unfixes the age of interaction so it

again will be defined by the moment of the last event.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.setInteractionAge( _data.ixnid, true );
```

NOTE: **This method currently applies to voice interactions only.** For multimedia interactions the following can be employed to provide the equivalent information. Do not apply the following to voice interactions.

To set the interaction age at the time that the interaction is present, similar to keep set to true, for example within a script on the entry to the next state:

```
var timestamp = Math.floor((new Date().getTime())/1000 ) + ' 0';
var setIXNAge = { RouterData70 : '("t"="' + timestamp + '")' };
_genesys.ixn.setuData(setIXNAge);
```

For the equivalent of keep set to false:

```
   _genesys.ixn.deleteuData( "RouterData70" );
```

_genesys.queue.setAdjustment

This function enforces an adjustment of the values of a specified statistic for a particular target. The adjustment does not apply to the result of explicit Stat Server queries by the function's `_genesys.statistic.sData()` function. It is only used for thresholds, statistical interaction distribution, or when a statistic is supplied as an argument to the `<submit>` actions. Once set, an adjustment can be cleared by invoking the _genesys.queue.resetAdjustment function. This applies across all interactions and the corresponding `<submit>` actions for the session.

`void _genesys.queue.setAdjustment(ixnid, target, statistic, sign, value)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **target:** STRING which can be a variable or a constant - This parameter defines the target whose statistic adjustment is to be set. The string value of this parameter must be in the possible target format. See the Target Formats section for details.
- **statistic:** STRING which can be a variable or a constant - This parameter defines the name of the statistic which is to be adjusted.
- **sign:** STRING which can be a variable or a constant - This parameter defines the operation that will be applied to adjust the statistic. The following are the valid values:
  - +, the value is added to the result reported by Stat Server
  - -, the value is subtracted from the result reported by Stat Server
    - , the value is multiplied by the result reported by Stat Server
  - /, the result reported by Stat Server is divided by value.

- **value:** NUMBER (FLOAT) which can be a variable or a constant - This parameter defines the value that the statistic is to be adjusted by.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.setAdjustment( _data.ixnid, 'SipGr_2.GA', 'StatAgentsAvailable', '*', 2 );
```

_genesys.queue.translationOverride

This function overrides any translation specified in configured Switch Access Codes for routing to remote targets later in the session, and instructs the functional module to use the information specified in the parameters for the purpose of number translation.

```
void _genesys.queue.translationOverride(ixnid, source, destination, location, rType,
DNIS, reason, extension)
```

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **source:** STRING which can be a variable or a constant - This parameter is the source device address to be used in the translation.
- **destination:** STRING which can be a variable or a constant - This parameter is the destination device address to be used in the translation.
- **location:** STRING which can be a variable or a constant - This parameter is the location of the source device to be used in the translation.
- **rType:** genesys.queue.rType ENUM Object which can be a variable or a constant - This parameter is the route type to be used in the translation.
- **DNIS:** STRING which can be a variable or a constant - This parameter is the DNIS to be used in the translation.
- **reason:** STRING which can be a variable or a constant - This parameter is the routing reason to be used in the translation.
- **extension:** STRING which can be a variable or a constant - This parameter is the extension information to be used in the translation.

Returns: `void`

The following is an example of input and output for this function:

```
_genesys.queue.translationOverride( _data.ixnid, 'source', 'dest', 'location', 1, '702',
'reason', 'ext' );
```

_genesys.queue.targetSelectionTuning

This function activates a configured cost-based routing solution for this session and the associated functional module objects (interactions and so on).

```
void _genesys.queue.targetSelectionTuning(ixnid, useCostFactor)
```

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **useCostFactor:** BOOLEAN which can be a variable or a constant - This parameter indicates whether cost factors should be used for making routing decisions for this session and the `<submit>` actions.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.targetSelectionTuning( _data.ixnid, true );
```

_genesys.queue.useAgentState

This function specifies the agent state the functional module will use instead of the default reported by Stat Server. This will apply to the entire session and all the `<submit>` actions until the next _genesys.queue.useAgentState invocation.

To use this option you must first set up URS as follows:

- In the *Annex* or *Options* tab of the platform application, create a section called AgentStates (case sensitive).
- Within that section, create an option for every user-defined agent state. The platform can accept up to 32 options in the *AgentStates* section.
- For each option, specify its value in the format:

```
Function[DNtype]<op1>Function[DNtype]<op1>...Function[DNtype]<op2>number...(Format1
expression)<op3>(Format1 expression)<op3>...
```

Where:

- `Function[DN type]` is one of the following predefined functions:
  - `ready[DN type]` - which returns the number of agent DNs of the specified type who are in the ready state at the current moment.
  - `busy [DN type]` - which returns the number of agent DNs of the specified type who are in the busy state at the current moment.

DN type for these predefined functions is the agent's DN. Types include *ACDPosition, Extension, E-mail, Eaport, Cellular, Chat, Cobrowse, Fax, Voicemail, Voip, Video*, and *Workflow*.

- op1 is an operator of either plus (+) or minus (-)
- op2 is an operator of either greater than (> - >), less than (< - <), or equal to (=)
- op3 is a logical operator of either or (|) or and (& - &)
- multiplication (*)
- division (/)
- number is zero or any positive number to evaluate the expression

For example, the agent is defined as ready if the agent has no calls on his or her extension or `position busy[extension]+busy[acdposition] = 0`. An agent in this state will be considered not ready if the agent has at least one call on the extension or position. The agent will be considered ready in all other situations. So, for example, if the agent has the e-mail DN busy, the agent is still considered ready.

Important Information

- When using the UseAgentState function, whole numbers are rounded (`1.25` is counted as `1`).
- Option values cannot contain spaces.
- If you want to use AgentStates for a backup functional module platform in addition to the primary platform, create an identical AgentStates section in the backup platform.
- When using this function, you must use lowercase DN types that do not include a "-" (hyphen). For example, use "email" instead of "E-mail".
- If function CheckAgentState is set to false, the functional module ignores any agent state, whether the default one (reported by Stat Server) or the user-defined one (as described above).
- The functional module uses integer arithmetic in its calculations, such as for agent state and skill expression evaluation. For this reason, you must always create expressions based on integer arithmetic, not floating point arithmetic.

`void _genesys.queue.useAgentState(ixnid, state)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **state:** STRING which can be a variable or a constant - This parameter is the agent state that should be used instead of the default reported from Stat Server.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.useAgentState( _data.ixnid, 'ready' );
```

_genesys.queue.useAgentStatistics

This function makes the functional module apply statistics for target selection at the level of individual Agents or Places even if the targets are groups of corresponding objects, such as Agent Groups or Place Groups.

This function may be used for any appropriate statistic. It can also be used for cost-based routing.

`void _genesys.queue.useAgentStatistics(ixnid, use)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **use:** BOOLEAN which can be a variable or a constant - This parameter indicates whether individual

agent statistics should be applied. The default is false.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.useAgentStatistics( _data.ixnid, true );
```

_genesys.queue.useCapacity

This function instructs the functional module on the condition for using configured capacity tables for computing statistical values. If *OnStatError* is supplied to the parameter, the functional module will compute a statistical value from statistical tables whenever an attempt to obtain the corresponding value from Stat Server results in an error. The other two options are either always use statistical tables for such values or never use them.

`void _genesys.queue.useCapacity(ixnid, use)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **use:** genesys.queue.usecapcond ENUM OBJECT which can be a variable or a constant - This parameter indicates the method of computing statistical values.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.useCapacity( _data.ixnid, 1 );
```

_genesys.queue.useCustomType

This function instructs the functional module on the value of a target property that the target must to have to be considered a valid target for the current session and the associated interaction and the `<submit>` actions that are being used.

`void _genesys.queue.useCustomType(ixnid, tType, property, value)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **tType:** STRING which can be a variable or a constant - This parameter indicates the type of target to use. The following are the valid values:
  - "Agent" - An agent target
  - "DN" - A DN target
  - "Place" - A Place target
- **property:** STRING which can be a variable or a constant - This parameter indicates which property of the target will be used. The following is where these properties are defined and found:

    - "Agent" target type - The property is specified under the Switch object, inside the folder with the name of the platform application, in the Annex of the Agent Login. The functional module checks the agent type for the following targets: *Agents, Agent Groups, Places,* and *Place Groups.* For the last two targets, the agent type can be verified only if Stat Server reports the name of the agent associated with the Place in question.
    - "DN" target type - The property is specified under the Switch object, inside the folder with the name of the platform application, in the Annex of the DN. The functional module checks the DN type for the following targets: *Agents, Agent Groups, Places, Place Groups, Routing Points*, and *Queues*, as well as custom *DNs*.
    - "Place" target type - The property is specified in the Annex of Place inside the folder with the name of the platform application. The functional module checks the Place type for the following targets: *Agents, Agent Groups, Places*, and *Place Groups*.
- **value:** STRING which can be a variable or a constant - This parameter indicates the value of the target property which should be evaluated to determine if the target should be used for routing.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.useCustomType( _data.ixnid, 'DN', 'Extension', '702' );
```

_genesys.queue.useDNType

This function instructs the functional module on the type of DN to use when routing to a target. Choose the DN-based type of session and interaction being routed. This delivers various sessions and interactions to the correct DN on an agent's desktop.

Important information

- The functional module sets the default DN type from the *MediaType* attribute of the trigger event. This function overrides that default.

`void _genesys.queue.useDNType(ixnid, type)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **type:** genesys.resource.resourceType ENUM OBJECT which can be a variable or a constant - This parameter indicates the type of DN target to use.

Returns: `void`

The following is an example of input and output for this function:

```
_genesys.queue.useDNType( _data.ixnid, _genesys.resource.resourceType.CFGACDPosition );
```

_genesys.queue.useMediaType

This function instructs the functional module on the media types (for examples, voice, e-mail, or chat) that a target is associated with. In order to accept sessions and interactions of a particular media type, a target must be associated with that media type. If the interaction is not a voice interaction, the functional module sets the initial media type from the Media Type attribute of the trigger event. The UseMediaType function overrides this initial setting. The functional module can pick up for routing either the available media of an agent or a ready DN of an agent.

Important Information

- For backward compatibility, if some DN type (DNTYPE) has a corresponding media (MEDIA) associated with it (for example e-mail), then the `_genesys.queue.useMediaType` function is equivalent to `_genesys.queue.useDNType` function.
- This function indicates the functional module will select only Extensions or ACDPositions of the agent.

`void _genesys.queue.useMediaType(ixnid, type)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **type:** STRING which can be a variable or a constant - This parameter indicates the type of Media target to use. The set of valid values are from the media definitions in the configuration layer.

Returns: `void`

The following is an example of this function:

```
_genesys.queue.useMediaType( _data.ixnid, 'CFGEmail' );
```

_genesys.queue.expandGroup

This function creates a list of resources associated with a group. This list of (agent) resources can be used for the following purposes:

- To propagate a target-selecting statistic on the resource level.
- To allow the Queue functional module to handle situations in which a particular resource is a member of multiple groups and to solve related interaction-priority issues.

`resourcelist _genesys.queue.expandGroup(ixnid, group)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **group:** STRING which can be a variable or a constant - This parameter is a string in the Target Formats. It will be the name of an Agent Group (real or virtual) or a Place Group (`groupname@location.GA or groupname@location.GP`) **Note:** location is optional, type is mandatory.

Returns: `resourcelist`: STRING - The result of the function is a string which contains a comma-separated list of Agents or Place resources belonging to the specified Agent Group or Place Group. For example, `"agent1@StatServer1.A,...,agentN@StatServer1.A"`

The following is an example of input and output for this function:

```
_data.myResult = _genesys.queue.expandGroup( _data.ixnid, 'SipGr_2.GA' );
```

_data.myResult will be the following string:

```
'701_sip@StatServer.A,702_sip@StatServer.A'
```

_genesys.queue.getSkillInGroup

This function returns the list of the agents belonging to the Agent Group based on the defined skill expression. You can also just specify a Stat Server and a skill expression without specifying an Agent Group. However, a skill expression and a Stat Server must be specified. The Stat Server is used to query the content of the provided Agent Group (real or virtual).

`resourcelist _genesys.queue.getSkillInGroup(ixnid, statserver, group, sexpr)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **statserver:** STRING which can be a variable or a constant - This parameter is optional. It is the name of the Stat Server containing information on the agents for this function. If not supplied, the Stat Server used will be the one defined by the *default_stat_server* configuration option of the platform.
- **group:** STRING which can be a variable or a constant - This parameter is optional. It is the agent group for the Stat Server that this function checks against. It can either be an agent group name or a list of comma-separated lists of agents, agent groups, places or place groups. Agents are included this group by either placing the agent name from the Persons folder into the Agent Groups folder or defining a virtual group using skill expression within the Annex tab of the Agent Group object.
- **sexpr:** STRING which can be a variable or a constant - This parameter defines the skill expression used to evaluate the agents. It can use skills, variables, numeric constants, and statistics to filter out agents based on their state. The statistic name in a skill expression can be any agent statistic used in the function `SData`. It must be written in the format: $(statistic). This ability to use statistics in a skill expression allows you to conduct queries based on a statistic. For example, if you want to query the number of agents with a Spanish skill of at least 5 who are currently logged in, the expression would be as follows: `Spanish >= 5 & $(StatAgentsLoggedIn)=1`.

Returns: `resourcelist`: String - The result of the function is a string which contains a comma-separated list of Agents or Place resources that meet the conditions of the skill expression. For example, `"agent1@StatServer1.A,...,agentN@StatServer1.A"`

The following is an example of input and output for this function:

```
_data.myResult = _genesys.queue.getSkillInGroup( _data.ixnid, 'StatServer', 'SipGr_2',
'english>3' );
```

_data.myResult will be the following string:

```
'701_sip@StatServer.A,702_sip@StatServer.A'
```

_genesys.queue.expandActivity

This function creates a list of resources associated with a workforce management activity. It is intended for use with Genesys Workforce Management. It takes as a parameter a Workforce Management Activity name (defined in the configuration layer) and returns the list of resources assigned to the Activity from the current moment up to the next CutOffTime number of seconds. If a resource assigned to the Activity in the given time interval has a break of any kind (including assignment to another activity), that resource will not be included in the returned list.

`resourcelist _genesys.queue.expandActivity(ixnid, activity, cutoff)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.
- **activity:** STRING which can be a variable or a constant - This parameter is a string. It will be the name of the workforce management activity to be used.
- **cutoff:** NUMBER which can be a variable or a constant - This parameter is the number representing the time in seconds that a resource has to be assigned to the activity starting from the current moment in order to be considered as qualified.

Returns: `resourcelist`: STRING - The result of the function is a string which contains a comma-separated list of agent resources belonging to the specified activity and meeting the cutoff time criteria. For example, `"agent1@StatServer1.A,...,agentN@StatServer1.A"` If the specified activity is not found, a null will be returned.

The following is an example of input and output for this function:

```
_data.myResult = _genesys.queue.expandActivity( _data.ixnid, 'taskA', 3 );
```

_data.myResult will be the following string:

```
'702_sip@StatServer.A'
```

_genesys.queue.resetTreatments (since 8.1.200.00)

This function clears the treatments associated with all outstanding `<submit>` actions associated with this session and can suspend the `<submit>` processing for the current session until the appropriate state has been reached to begin the next treatments.

When a list of busy treatments is used in a `<queue:submit>`, Universal Routing Server uses this list and plays the busy treatments one after another (by default, this list loops around). When the function `_genesys.queue.resetTreatments` is sent to Universal Routing Server, it will clear the list of busy treatments. However, if a treatment was being played while this function is received, the treatment currently playing will not be cleared. It is recommended to disable looping using `repeat` in the resource attribute for `<dialog:playsound>`.

`void _genesys.queue.resttreatments(ixnid)`

Parameters:

- **ixnid:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have this action applied.

Returns: `void`

The following is an example of this function being used:

```
<state id="findAgent">
        <onentry>
                <queue:submit priority="5" timeout="60">
                        <queue:targets>
                                <queue:target type="agent" name="'agent1'" />
                        </queue:targets>
                        <dialog:runtreatments>
                                <dialog:playsound type="'music'" resource="'music/
on_hold;repeat=1'" duration="15" />
                                <dialog:playsound type="'music'" resource="'music/
in_queue;repeat=1'" duration="15" />
                        </dialog:runtreatments>
                </queue:submit>
        </onentry>
        <transition event="stopMusic" target="resetTreatments" />
        <transition event="queue.submit.done" target="state2" />
        <transition event="error.queue.submit" target="state3" />
</state>
<state id="resetTreatments">
        <onentry>
                <script>
                        _data.myResult = _genesys.queue.resetTreatments(_data.ixnid);
                </script>
        </onentry>
</state>
```

If the `stopMusic` event is received within the first 15 seconds while the `music/on_hold` is playing, it will play until the end of the file (ignoring the duration attribute), and `music/in_queue` will never be played because `resetTreatments` has cleared it from the list.

If the `stopMusic` event is received 50 seconds after entering the state `findAgent`, `music/in_queue` will be playing when the event is received, and the file will play until the end (igoring the duration attribute). No other music file will be played after this.

_genesys.queue.setDNIS

This function replaces the DNIS of the interaction by the appropriate value. It will override any previous`_genesys.queue.setDNIS()` function invocations.

Important Note: In order to get this DNIS override into the actual interaction on the controlling server (T-Server), the session- and interaction-originating event (for example, `EventRouteRequest`) must have had the routing type of *RouteTypeOverwriteDNIS* to make the change in the server. Otherwise, the override is only set within the session and its associated interaction object.

`void _genesys.queue.setDNIS(ixn, type, value)`

Parameters:

- **ixn:** STRING which can be a variable or a constant - This parameter is mandatory. It defines the ID of the interaction which should have its DNIS overridden.
- **type:** genesys.queue.overwriteType ENUM OBJECT which can be a variable or a constant - This parameter defines the source of the value to set the DNIS to.
- **value:** STRING which can be a variable or a constant - This parameter defines the value that the DNIS is to be set to. This parameter is only valid when the type parameter is set to "UseValue".

Returns: `void`

The following is an example of input and output for this function:

```
_genesys.queue.setDNIS( _data.ixnid, 2, '3318' );
```

_genesys.queue.useMediaChannel (since ORS 8.1.200.48 and URS 8.1.200.25)

This function provides information about the type of controller that owns the interaction (either T-Server/SIP-Server or Interaction Server). In most cases, Queue FM itself defines the right controller type and there is no need to use this function explicitly. However, some types of media interactions (like chats) can be managed by either type of controller and Queue FM needs to be made aware of the exact controller type.

If Queue FM is called in "interactionless" context (when ixnid is set to null), Queue FM assumes T-Server/SIP-Server as the default controller type. If this is not the case, function `useMediaChannel` can be used to change this default.

`void _genesys.queue.useMediaChannel(ixnid, enable)`

Parameters (all parameters are mandatory):

- **ixnid:** STRING which can be a variable or a constant - This parameter defines the ID of the interaction which should have this action applied.
- **enable:** BOOLEAN which can be a variable or a constant - This parameter defines whether the interaction is controlled by T-Server/SIP-Server (if it is set to false) or Interaction Server (if it is set to true).

Returns: `void`

The following is an example of this function. When an SCXML session is started through Web API and Queue FM is used to find an agent ready to accept chats controlled by Interaction Server, the following should be invoked:

```
_genesys.queue.useMediaChannel(null, true);
```

Parameter Elements

The `<submit>` action element has parameter elements that can be used as input for the target and outbound attributes.

<targets>

This is the top-level element which defines the set of targets from which a given target is selected for the submit request.

Attributes

Name	Required	Type	Default Value	Valid Values	Description
type	false	NMTOKEN	none	agent, place, agentgroup, placegroup, queue, dn, skill, campaigngroup, routepoint, label	This specifies the default resource type that should be used if a type attribute is not specified in the associated `<target>` elements.
statserver	false	value expression	none	any value expression that returns a valid string	A value expression which returns the default statserver that should be used if a statserver attribute is not specified in the associated `<target>` elements. See SCXML Legal Data Values and Value Expressions for details.

Children

- `<target>` Occurs 0 to N
- `<targetid>` Occurs 0 to N
- `<activity>` Occurs 0 to 1 - This element is also mutually exclusive with the other child elements.
- `<workbin>` Occurs 0 to 1 - This element is used as the target's workbin of objects specified in the `<targets>` element.

<targetid>

This defines a specific means of representing a set of targets.

Attributes

Name	Required	Type	Default Value	Valid Values	Description
expr	true	value expression	none	Any value expression that returns a string that follows the format defined in the description	This is the ID of the target that is to be used. It is a string with a set of comma-separated sub-strings with the following format:

Name	Required	Type	Default Value	Valid Values	Description

					`{threshold}[weight]name@server.t` or `{threshold}[weight]?name:skillex`
					
					 The following is an example:
					
					 `"'{StatCallsInQueue <10}[25]8001.Q, {StatCallsInQueue < 10}[25]8002.Q, {StatCallsInQueue < 20}[50]8003.Q"'`
					
					 The following are the details of the different elements of the sub-strings.
					
					 • weight - same as defined in `<target>` element
					 • name - same as defined in `<target>` element
					 • server - same as defined in statserver attribute in the `<target>` element
					 • type - same as defined in `<target>` element, but with the following abbreviations:
					   • A (for agent),
					   • AP (for agent place)
					   • GA (for group of

Name	Required	Type	Default Value	Valid Values	Description
					agents) • GP (for group of places) • Q (for Queues) • DN (for dn) • GA (for skill) • GC (for campaign group) • RP (for routing points) • DL (for destination label) • skilexpr - same as defined in `<target>` element • threshold - same as defined in `<target>` element See SCXML Legal Data Values and Value Expressions for details.

Children

None

<target>

This defines the resource criteria for selecting a target.

Attributes

Name	Required	Type	Default Value	Valid Values	Description
name	false	value expression	none	Any value expression that returns a valid string which represents a resource of the defined type.	A value expression which returns the name of the target that is to be used. See SCXML Legal Data Values and Value Expressions for details.
skillexpr	false	value expression	none	Any value expression that returns a valid string which represents a valid skill expression	This is the skill expression associated with this `<target>` element. For details on the format, see the Skill Expressions section. See SCXML Legal Data Values and Value Expressions for details.
type	false	NMTOKEN	none	agent, place, agentgroup, placegroup, queue, dn, skill, campaigngroup, routepoint, label	This specifies the resource type associated with this `<target>` element. The skill and agentgroup types are synonymous.
statserver	false	value expression	none	Any value expression that returns a valid string which represents a valid stat server.	A value expression which returns the statserver that should be used for this target definition. See SCXML Legal Data Values and Value Expressions for details.
threshold	false	value expression	none	Any value expression that returns a valid string which represents a valid threshold expression	A value expression which returns a criteria definition that is used to further filter the potential possible targets associated with this `<target>` attribute. threshold is an analog of the strategy function

Name	Required	Type	Default Value	Valid Values	Description
					SetTargetThreshold and defines additional conditions the target must meet to be considered as valid target for routing. The following queue-specific methods can be used in a value expression. These methods are not executed inline as part of interpreting this attribute but are processed by the underlying queue functional module: • **`sdata(target, statistic)`**- Use this function to affect routing conditions based on statistics. • **`Acfgdata(Application name, folder, property, default value)`** - Use this function to affect routing conditions based on external data stored in properties of configuration layer application objects (ApplicationConFigDATA). • **`Lcfgdata(list name, folder, property,`**

Name	Required	Type	Default Value	Valid Values	Description
					`default value)` - Use this function to affect routing conditions based on external data stored in IRD list objects. • `callage[]` - Use this function to return the age of an interaction in seconds. See SCXML Legal Data Values and Value Expressions for details. **Note:** Orchestration evaluates this expression to create a string that is sent to Universal Routing Server. That is, variables and functions available in ORS may be used to construct this string; however URS has no notion of these values. The resultant string must be a completely self contained expression for URS. In other words, if you have a variable *x* in your strategy, and you want to use callage[] < *x*, it should be written as: 'callage[] < ' + *x*.
weight	false	value expression	none	Any value expression that returns a valid string	A value expression which returns a value that defines the weight of this `<target>` element against other

Name	Required	Type	Default Value	Valid Values	Description
					`<target>` elements. See SCXML Legal Data Values and Value Expressions for details.

- The name, skillexpr, weight, statserver and type attributes are interpreted as constants. They are components of the target specification format `[weight]name@statserver.type`. If skillexpr is present, then the format is `[weight]?name:skillexpression@statserver.type`.

Children

None

<activity>

This defines the workforce management activity criteria for selecting a target.

Attributes

Name	Required	Type	Default Value	Valid Values	Description
name	true	value expression	none	Any value expression that returns a valid string which represents the name of a WFM activity	A value expression which returns the name of a WFM activity that is to be used for target selection. See SCXML Legal Data Values and Value Expressions for details.
cutofftime	false	value expression	none	Any value expression that returns an integer	A value expression which returns the cutoff time in seconds. This defines the window time in which a potential target resource must be assigned to the given activity. See SCXML Legal Data Values and Value Expressions for details.

Children

None

<workbin>

This defines the workbin criteria for selecting a target.

Attributes

Name	Required	Type	Default Value	Valid Values	Description
type	true	NMTOKEN	none	agent, place, agentgroup, placegroup	This specifies the resource type associated `<workbin>` element.
name	true	value expression	none	Any value expression that returns a valid string which represents a workbin.	A value expression which returns the name of the workbin that is to be used. See SCXML Legal Data Values and Value Expressions for details.
loggedinonly	false	boolean expression	false	Any value expression that returns true or false	A boolean expression which returns whether logged out agents can pull interactions. (Dis)Allow using of logged out agents.

Children

None

Action Elements

<submit>

This action queues the request for a target based on the criteria specified in the request.

Attribute Details

Name	Required	Type	Default Value	Valid Values	Description
requestid	false	location expression	none	Any valid location expression	This is the location for the request ID that is returned as part of this request. Any data model expression evaluating to a data model location. See SCXML Location Expressions for details. The location's value will be set to an internally generated integer identifier to be associated with the action being sent. This value will only be valid when the queue.submit.requestid event is received. If this attribute is not specified, the event identifier is dropped. This identifier can be tested by the completion event handler to distinguish among several outstanding requests. If this attribute is not specified, the identifier can be acquired from the action completion event. Every request must receive a unique identifier.
queue	false	value expression	none	Any value expression which returns a valid string	A value expression which returns the name of the (virtual) queue that this request should be put in. See SCXML Legal Data Values and Value Expressions for details.
priority	false	value expression	0	Any value expression which returns a value integer	A value expression which returns the priority that the interaction will be

Name	Required	Type	Default Value	Valid Values	Description
					given in the queue. See SCXML Legal Data Values and Value Expressions for details.
ordertype	false	value expression	any	max, min, any, percentage	A value expression which returns how type of ordering that should be used on the targets. It is used together with **orderstat**. See SCXML Legal Data Values and Value Expressions for details.
orderstat	false	value expression	none	Any value expression which returns a valid string	A value expression which returns the name of the statistic that will be used as a target selection criterion for ordering the targets. No orderstat specified means any target and order. See SCXML Legal Data Values and Value Expressions for details.
interactionid	false	value expression	none	Any value expression which returns a valid string	A value expression which returns the _genesys.ixn.interactions[x].g_uid associated with this request. There is a special value that can be returned: • ECMAScript **Null** means the functional module will not use an interaction for the request. **Note:** if the `<outbound>` element is present then this attribute

Name	Required	Type	Default Value	Valid Values	Description
					is ignored because a new interaction will be created. See SCXML Legal Data Values and Value Expressions for details.
route	false	boolean expression	true	Any expression which returns a boolean (true, false)	A boolean expression which returns whether or not this action should also redirect the interaction to the selected destination. There are two values that can be returned: • **"false"** means the functional module will not attempt to route the associated interaction. • **"true"** means the functional module will use the associated interaction to route the interaction. **Note:** a value of "true" is only supported for voice-related interactions. See SCXML Conditional Expressions for details.
clearontimeout	false	boolean expression	true	Any expression which returns a boolean (true, false)	A boolean expression which indicates whether or not the request and all associated

Name	Required	Type	Default Value	Valid Values	Description
					`<submit>` requests for this interaction and queue should be removed from the queue after the timeout of this request. See SCXML Conditional Expressions for details.
timeout	false	value expression	0	A value expression which returns an integer	A value expression which returns an integer that represents the number of seconds to wait. See SCXML Legal Data Values and Value Expressions for details. The integer returned must be interpreted as a time interval. This interval begins when `<submit>` is executed. A failed and timed-out submit must return the error.queue.submit event. If the `<outbound>` element is present, this attribute is only used while waiting for an agent to become available and not for the request in its entirety.
threshold	false	value expression	none	Any value expression that returns a valid string which represents a valid threshold expression.	A value expression which returns a criteria definition that is used to further filter the potential possible targets associated with the queue attribute. **threshold** is an analog of the strategy function SetVQTargetThreshold and defines additional conditions the target must meet to be considered

Name	Required	Type	Default Value	Valid Values	Description
					as a valid target for routing with this queue. The following queue-specific methods can be used in a value expression. These methods are not executed inline as part of interpreting this attribute but are processed by the underlying queue functional module: • **sdata**(target, statistic) - Use this function to affect routing conditions based on statistics. • **acfgdata**(Application name, folder, property, default value) - Use this function to affect routing conditions based on external data stored in properties of configuration layer application objects (ApplicationConFigDATA). • **lcfgdata**(list name, folder, property, default value) - Use this function to

Name	Required	Type	Default Value	Valid Values	Description
					affect routing conditions based on external data stored in IRD list objects. • **callage** function - Use this function to return the age of an interaction in seconds. See SCXML Legal Data Values and Value Expressions for details.
src	false	value expression	none	A value expression which returns one of the following valid URI schemes: • gdata	This allows a developer to supply a URI which identifies the location of a `<submit>` definition that is to be used in the application. This attribute is mutually exclusive with the following attributes: • queue • ordertype • orderstat • timeout This attribute is also mutually exclusive with the children of this element. This source definition will replace the entire `<submit>` element in the application at load time. See SCXML Legal Data Values and Value Expressions

Name	Required	Type	Default Value	Valid Values	Description
					for details.

Other Considerations

- All events generated from the imbedded treatment actions will be suppressed as part of the action.
- The `<submit>` action does not work for targets such as multimedia queues. The `<redirect>` action is a way to place interactions into those queues.
- The **src** attribute supports only the gdata scheme, which maps to routing rules of type:
  - Statistic
  - Load Balance
  - Percentage
  - Workforce
- The route=true attribute value is supported only for voice-related interactions. For all others, route= false must be used with a subsequent `<redirect>` action. The error.queue.submit will be generated if this attribute is implicitly or explicitly set to true for all other interaction media types.
- The following rules are applied to the interactionid parameter in this action as well as in any other from the Queue functional module:
  - Genesys recommends that you always specify it explicitly.
  - If the `<scxml>` document has _type=routingstrategy and there are no interactions associated with the session (for example, a session is started through web interfaces), the special value null can be used. This will result in the Queue functional module using a platform-specific interactionid for communication purposes.

The following are examples of different types of target selection that can be done with `<submit>` and SCXML:

- **Load Balancing**

As you can see if you want to perform load balance functionality similar to the load balancing function block, you would use one of the load balancing statistics (this is using the basic one). In addition, you can really only load balance on DN-related resources (queue and dn types).

```
<state id="load_balancing">
<datamodel>
  <data id="reqid"/>
</datamodel>
<onentry>
  <queue:submit requestid="_data.reqid" orderstat="'StatLoadBalance'" ordertype="'min'"
timeout="100" >
    <queue:targets type="queue">
      <queue:target name="'queue1'"/>
      <queue:target name="'queue2'"/>
    </queue:targets>
  </queue:submit>
</onentry>
<transition event="queue.submit.done" target="statex">
```

```
  <assign location="target" expr="_event.data.targetselected"/>
</transition>
<transition event="error.queue.submit" target="statey"/>
</state>
```

- **Percentage**

As you can see, if you want to perform percentage routing functionality similar to the percentage function block, you set the ordertype attribute to "percentage" and set the appropriate target weights.

```
<state id="percentage">
<datamodel>
  <data id="reqid"/>
</datamodel>
<onentry>
  <queue:submit requestid="_data.reqid" ordertype="'percentage'" timeout="100" >
    <queue:targets type="agentgroup">
      <queue:target name="'agtgrp1'" weight="'20'"/>
      <queue:target name="'agtgrp2'" weight="'80'"/>
    </queue:targets>
  </queue:submit>
</onentry>
<transition event="queue.submit.done" target="statex"/>
<transition event="error.queue.submit" target="statey"/>
</state>
```

- **Selection (Generic Target Selection)**

This type of target selection is generic and can be used to combine several different forms of target selections. The biggest difference is the ability to define thresholds for a given target definition. This example shows how to do that.

```
<state id="Selection">
<datamodel>
  <data id="reqid"/>
</datamodel>
<onentry>
  <queue:submit queue="'VQ1'" requestid="_data.reqid" ordertype="'max'"
orderstat="'StatTimeInReadyState'" timeout="100">
    <queue:targets type="agentgroup" statserver="'www.genesyslab.stserver1.com'">
      <queue:target name="'agtgrp1'" threshold= "'sdata(agtgrp1.GA, StatAgentsAvailable) >
10'"/>
      <queue:target name="'agtgrp2'"/>
    </queue:targets>
  </queue:submit>
</onentry>
<transition event="queue.submit.done" target="statex"/>
<transition event="error.queue.submit" target="statey"/>
</state>
```

- **Statistics**

As you can see here, if you want to perform statistical routing functionality similar to the statistics function block, you just have to specify the ordertype and orderstat on the action and either specify a single statserver for all targets or a specific one for each to do target selection.

```
<state id="Statistics">
<datamodel>
```

```
  <data id="reqid"/>
</datamodel>
<onentry>
  <queue:submit queue="'VQ1'" requestid="_data.reqid" ordertype="'min'"
orderstat="'StatExpectedWaitingTime'" timeout="100">
    <queue:targets type="agentgroup" statserver="'www.genesyslab.stserver1.com'">
      <queue:target name="'agtgrp1'"/>
      <queue:target name="'agtgrp2'"/>
    </queue:targets>
  </queue:submit>
</onentry>
<transition event="queue.submit.done" target="statex"/>
<transition event="error.queue.submit" target="statey"/>
</state>
```

- **Skills-Based**

As you can see here, if you want to perform skills-based routing functionality similar to that which is available on several function blocks, you just have to specify the skillexpr attribute on a target element. This can be mixed with other targets as well.

```
<state id="Skills-based">
<datamodel>
  <data id="reqid"/>
</datamodel>
<onentry>
  <queue:submit queue="'VQ1'" requestid="_data.reqid" ordertype="'min'"
orderstat="'StatExpectedWaitingTime'" timeout="100">
    <queue:targets type="agentgroup" statserver="'www.genesyslab.stserver1.com'">
      <queue:target skillexpr="'service1 > 5 & english > 3'"/>
      <queue:target name="'agtgrp2'"/>
    </queue:targets>
  </queue:submit>
</onentry>
<transition event="queue.submit.done" target="statex"/>
<transition event="error.queue.submit" target="statey"/>
</state>
```

- **Using Busy Treatments**

As you can see here, if you want to perform treatments in conjunction with routing functionality you need to specify the appropriate `<runtreatments>` element defining which treatments are to run while waiting for a target to be found.

```
<state id="Skills-based_with_treatments">
<datamodel>
  <data id="reqid"/>
</datamodel>
<onentry>
  <queue:submit queue="'VQ1'" requestid="_data.reqid" ordertype="'min'"
orderstat="'StatExpectedWaitingTime'" timeout="100">
    <queue:targets type="agentgroup" statserver="'www.genesyslab.stserver1.com'">
      <queue:target skillexpr="'service1 > 5 & english > 3'"/>
      <queue:target name="'agtgrp2'"/>
    </queue:targets>
    <dialog:runtreatments>
      <dialog:playsound type="'music'" resource="'EMusicDN'" duration="100"/>
      <dialog:play language="'English (US)'" >
        <dialog:prompts type="'tts'">
          <dialog:prompt interrupt="true" text="'The estimated wait time is' +
_genesys.statistic.sData('agtgrp2@.GA', 'StatExpectedWaitingTime')"/>
```

```
        </dialog:prompts>
      </dialog:play>
    </dialog:runtreatments>
  </queue:submit>
</onentry>
<transition event="queue.submit.done" target="statex"/>
<transition event="error.queue.submit" target="statey"/>
</state>
```

- **Activity-Based**

As you can see here, if you want to perform activity(WFM)-based routing functionality similar to that which is available on several function blocks, you just have to specify the `<activity>` element in the `<targets>` element.

```
<state id="Activity-based">
<datamodel>
  <data id="reqid"/>
</datamodel>
<onentry>
  <queue:submit queue="'VQ1'" requestid="_data.reqid" ordertype="'min'"
orderstat="'StatExpectedWaitingTime'" timeout="100">
    <queue:targets statserver="'www.genesyslab.stserver1.com'">
      <queue:activity name="'service1'"/>
    </queue:targets>
  </queue:submit>
</onentry>
<transition event="queue.submit.done" target="statex"/>
<transition event="error.queue.submit" target="statey"/>
</state>
```

- **Using Routing Rules From Configuration Server**

As you can see here, if you want to perform routing rule routing functionality similar to the existing routing function block, you just have to specify the src attribute using the gdata URI scheme.

```
<state id="RoutingRules_from_Configuration_Server">
<datamodel>
  <data id="reqid"/>
</datamodel>
<onentry>
  <queue:submit src="gdata:routingrule1"/>
</onentry>
<transition event="queue.submit.done" target="statex"/>
<transition event="error.queue.submit" target="statey"/>
</state>
```

Children

- `<targets>` - Occurs 1 time. This instance defines a named set of targets.
- `<runtreatments>` - Occurs 0 or 1 times. This instance defines a set of treatments to apply while this request is queued. The treatment-related events will not be generated as part of this request. The treatment can have an additional parameter, duration (if it doesn't have it yet). When the treatment expires, the next treatment in the definition will be applied. If treatments that collect digits are used, the application will only get the digits collected by the last treatment that collected the digits when this action has ended (successfully or unsuccessfully). These digits will be available to the application via the _event.data.digits property.

Events

The following events can be generated as part of this action:

- `queue.submit.done`
- `queue.submit.requestid`
- `error.queue.submit` - This event will be sent as a result of the following conditions:
  - A timeout of the request
  - Problems with the request itself.
  - The customer abandons the interaction
  - The platform failed to contact the customer based on the criteria specified.
  - The platform failed to contact the agent that was selected.
- `queue.cancelled` - This event will be sent either when a target was selected from another outstanding `<submit>` action, in which case it indicates that this `<submit>` action request was cancelled, or when a `<submit>` action request has been cancelled using the `<cancel>` action.
- `interaction.deleted` - This event will be sent when the customer abandons the interaction associated with this `<submit>` request.

Note: For every `<queue.submit>` action, one and only one event can be created with the reference id generated with this submit.action. Specifically:

- `queue.submit.done` - One of the targets from this queue was selected.
- `error.queue.submit` - The submit action fails to apply.
- `queue.cancel.done` - The interaction was explicitly requested to be removed specifically from the queue with this request id.
- `queue.cancelled` - The interaction was removed from the queue as a side effect of another successfully completed action.

<cancel>

This action removes the requests from the queue and from consideration as targets. This is equivalent to doing another `<submit>` with the clearqueue attribute set to true or to issuing the IRD function ClearTargets.

Attribute Details

Name	Required	Type	Default Value	Valid Values	Description
requestid	false	location expression	none	Any valid location expression	This is the location of the ID of the outstanding request which is to be canceled. Any data model expression

Name	Required	Type	Default Value	Valid Values	Description
					evaluating to a data model location. See SCXML Location Expressions for details.
queue	false	value expression	none	Any value expression which returns a valid string	A value expression which returns the name of the (virtual) queue that this request is for. If there was more than one `<submit>` request for the same interaction and this queue, then all these requests will be removed from this queue and from consideration as a target. See SCXML Legal Data Values and Value Expressions for details.
interactionid	false	value expression	none	Any value expression which returns a valid string	A value expression which returns the _genesys.ixn.interactions[x].g_uid associated with this request. This is only used in conjunction with the queue attribute. There is a special value that can be returned: • ECMAScript **Null** means the functional module will not use an interaction for the request. See SCXML Legal Data Values and Value Expressions for details.

The requestid and queue attributes are mutually exclusive. If the requestid is present, then just that request is cleared from the queue and from target selection.

If the queue is present, then all outstanding requests are cleared from the defined queue. As part of this processing, the appropriate queue.cancelled events will be fired for all requests that are cleared.

The interactionid attribute is only used in conjunction with the queue attribute and is only needed when your application is handling multiple interactions.

The following are some examples:

```
<state id="cancel">
<datamodel>
  <data id="reqid"/>
</datamodel>
<onentry>
  <queue:cancel requestid="_data.reqid"/>
</onentry>
<transition event="queue.cancel.done" target="statex"/>
<transition event="error.queue.cancel" target="statey"/>
</state>
<state id="cancel">
<onentry>
  <queue:cancel queue="'vq1'" />
</onentry>
<transition event="queue.cancel.done" target="statex"/>
<transition event="error.queue.cancel" target="statey"/>
</state>
<state id="cancel">
<datamodel>
  <data id="ixnid"/>
</datamodel>
<onentry>
  <queue:cancel queue="'vq1'" interactionid="_data.ixnid" />
</onentry>
<transition event="queue.cancel.done" target="statex"/>
<transition event="error.queue.cancel" target="statey"/>
</state>
```

Children

None

Events

The following events can be generated as part of this action:

- `queue.cancel.done`
- `error.queue.cancel`
- `queue.cancelled` - This event is sent for all requests that are cleared.

Note: The `queue.cancelled` events will be sent before the `queue.cancel.done`.

<update>

This action updates the criteria associated with an outstanding submit request.

Attribute Details

Name	Required	Type	Default Value	Valid Values	Description
requestid	True	location expression	none	Any valid location expression	This is the location of the ID of the outstanding request which is to be updated. Any data model expression evaluating to a data model location. See SCXML Location Expressions for details.
interactionid	False	value expression	none	Any value expression which returns a valid string	A value expression which returns the _genesys.ixn.interactions[x].g_uid associated with this request. If the interactionid attribute value is not associated with an outstanding and corresponding `<submit>` action, then an error event (error.queue.cancelled) will be generated. There is a special value that can be returned: • ECMAScript **Null** means the functional module will not use an interaction for the request. See SCXML Legal Data Values and Value Expressions for details.
priority	False	value expression	none	Any value expression that returns a valid integer	A value expression which returns the priority that the interaction will be assigned in the queue. See SCXML Legal Data Values and Value Expressions for details.

Name	Required	Type	Default Value	Valid Values	Description
timeout	False	value expression	none	A value expression which returns an integer	A value expression which returns an integer that represents the number of seconds to wait. See SCXML Legal Data Values and Value Expressions for details. The integer returned must be interpreted as a time interval. This interval begins when `<update>` is executed. A failed and timed-out submit must return the error.queue.update event. If the original submit request an outbound one, this attribute is only used while waiting for an agent to become available and not for the request in its entirety.

The following are examples:

This example updates the timeout value for this request:

```
<state id="update">
<datamodel>
  <data id="reqid"/>
  <data id="ixnid"/>
</datamodel>
<onentry>
  <queue:update requestid="_data.reqid" interactionid="_data.ixnid" timeout="300"/>
</onentry>
<transition event="queue.update.done" target="statex"/>
<transition event="error.queue.update" target="statey"/>
</state>
```

This example updates the priority for this request:

```
<state id="update">
<datamodel>
  <data id="reqid"/>
</datamodel>
<onentry>
  <queue:update requestid="_data.reqid" priority="6" />
</onentry>
<transition event="queue.update.done" target="statex"/>
<transition event="error.queue.update" target="statey"/>
</state>
```

Children

None

Events

The following events can be generated as part of this action:

- `queue.update.done`
- `error.queue.update`

<query>

This action queries the status of the request. It returns the following information in the `queue.query.done` event for a specific interaction:

- priority - This is the current priority of the request.
- positioninqueue - This is the current position of the request in the queue.
- Invqwaittime - This is the expected wait time for the queue in relationship to the request.

This action can be used in two cases:

- While `<submit>` requests are outstanding, to determine what to do next. For example, while the application waits for a target, if the customer presses the "I cannot wait anymore" number, this event is processed by the application (`<transition>`). The application queries the status of the request to determine what type of treatment to present to the customer. If the customer has been waiting 10 minutes and the queue depth has not changed then the application may offer the customer the option of having a callback set up for him or her (using a dialog or treatment).
- After a `<submit>` request has timed out, if the interaction was not cleared from target selection. The application may need to collect information on where this interaction stands with respect to it still being in the queue. This information would then be used to determine what to do next (for example, provide the customer with other options, do another `<submit>` but with different target selection criteria (other targets, a new queue, and so on).

Attribute Details

Name	Required	Type	Default Value	Valid Values	Description
requestid	true	location expression	none	Any valid location expression	This is the location of the ID of the outstanding request which is to be queried. Any data model expression evaluating to a data model location. See SCXML Location Expressions for details.

Name	Required	Type	Default Value	Valid Values	Description
interactionid	false	value expression	none	Any value expression which returns a valid string	A value expression which returns the _genesys.ixn.interactions[x].g_uid associated with this request. If the interactionid attribute value is not associated with the outstanding corresponding `<submit>` action, then an error event (error.queue.query) will be generated. There is a special value that can be returned: • ECMAScript **Null** means the functional module will not use an interaction for the request. See SCXML Legal Data Values and Value Expressions for details.

The interactionid attribute is only used when your application is handling multiple interactions.

The following is an example:

```
<state id="Query">
<onentry>
  <queue:query requestid="_data.reqid">
</onentry>
<transition event="queue.query.done" target="statex">
  <if cond="_event.data.positioninqueue > 200">
    <queue:update requestid="_data.reqid" priority="_event.data.priority + 5"/>
  </if>
</transition>
<transition event="error.queue.query" target="statey"/>
</state>
```

Children

None

Events

The following events can be generated as part of this action:

- `queue.query.done`
- `error.queue.query`

<default>

This action will redirect the interaction to its associated default destination or may optionally returns the configured default target address with out redirecting the interaction. This may be used when the orchestration logic cannot find a suitable destination to redirect the interaction to.

Attribute Details

Name	Required	Type	Default Value	Valid Values	Description
requestid	false	location expression	none	Any valid location expression	This is the location for the request ID that is returned as part of this request. Any data model expression evaluating to a data model location. See SCXML Location Expressions for details. The location's value will be set to an internally generated unique string identifier to be associated with the action being sent. This value will only be valid when the queue.default.requestid event is received. If this attribute is not specified, the event identifier is dropped. This identifier can be tested by the completion event handler to distinguish among several outstanding requests. If this attribute is not specified, the identifier can be acquired from the completion event. Every request must receive a unique identifier.
interactionid	false	value expression	none	A valid value	A value expression

Name	Required	Type	Default Value	Valid Values	Description
				expression	which returns the _genesys.ixn.interactions[x].g_uid associated with this request. There is a special value that can be returned: • ECMAScript **Null** means the functional module will not use an interaction for the request. See SCXML Legal Data Values and Value Expressions for details.
route	false	boolean expression	true	Any expression that returns a boolean (true, false)	A boolean expression which returns true or false. The meaning of which implies the following: • **"false"** means the functional module will not attempt to route the associated interaction and only the default destination associated will be returned in the `queue.default.done` event. • **"true"** means the functional module will use the associated interaction to route the

Name	Required	Type	Default Value	Valid Values	Description
					interaction. **Note:** a value of "true" is only supported for voice-related interactions. See SCXML Conditional Expressions for details.

The following is an example:

```
<state id="Default">
  <onentry>
    <queue:default/>
  </onentry>
  <transition event="queue.default.done" target="statex"/>
  <transition event="error.queue.default" target="statey"/>
</state>
```

Children

None

Events

The following events can be generated as part of this action:

- `queue.default.done`
- `queue.default.requestid`
- `error.queue.default`

Events

The following are the Queue action result events:

Event	Attributes	Description
queue.submit.done		This event indicates the success of the request and that a target has been selected.

Event	Attributes	Description
	requestid	This is the ID associated with the request.
	targetselected	This is the DN and the switch name of the target to which the interaction was routed or should be routed to definitively; the target format is (`Name@SwitchName.Type`).
	vqselected	This is the virtual queue that was selected.
	targetcomponentselected	This is the agent-level target to which the interaction was routed or should be routed to definitively. If the target specified in `<submit>` and selected for routing is of type Agent, Place, Queue, or Routing Point, this contains the target itself. If the desired target type is Agent Group, Place Group, or Queue Group, the function returns the agent, place, or queue from the corresponding group the interaction was sent to. The target format is (`Name@StatServerName.Type`).
	targetobjectselected	This is the high-level target (one that you specify in a `<submit>`) to which the interaction was routed or should be routed to definitively. If a skill expression is used, the function returns: `?:SkillExpression@statserver.GA` or even `?GroupName:SkillExpression@statserver.GA` The target format is (`Name@StatServerName.Type`).
	resource	This is an ECMAScript resource object which represents the target selected and can be used on any interaction-related action. See the Resource object for details.
	access	This attribute is optional and provided only if <submit> parameter route set to false and siwtch access code is defined between source and destination swithces for target type that match type of selected target. When present it is an ECMAScript object which represents switch access code and has following sert of properties: prefix, rtype, destination, location and dnis. Their values match to following switch access code fileds: Code, Route Type, Destination Source, Location Source and DNIS Source.

Event	Attributes	Description
queue.submit.requestid		This event provides the application with request ID for the given request that was invoked.
	requestid	This is the ID associated with the request from the orchestration application or the resource.
error.queue.submit		This indicates that an abnormal condition occurred while trying to perform the request. This event will be sent as a result of a timeout of the request, as well as problems with the request or interaction itself.
	requestid	This is the ID associated with the request.
	error	This is the type of error that occurred. The following are the possible values: • -001 Timeout • 0001 Unknown error • 0008 Routing done • 0013 Remote error • 0018 Unknown object • 0019 Translation failed
	description	if error parameter has value '0013 Remote error' then this field might contain additional error information as provided by call/interaction controller (for example by TServer). Format of this string field is 'ErrorNumber ErrorMessage'
queue.cancel.done		This event indicates the success of the clear request and that the request has been removed from the queue.
	requestid	This is the ID of the `<cancel>` request.
error.queue.cancel		This indicates that an error occurred while trying to perform the `<cancel>` request.
	requestid	This is the ID associated with the request.

Event	Attributes	Description
	error	This is the type of error that occurred.
	description	This is a more detailed description of the error.
queue.update.done		This event indicates that the request has been updated successfully.
	requestid	This is the ID of the `<update>` request.
error.queue.update		This indicates that an error occurred while trying to perform the `<update>` request.
	requestid	This is the ID associated with the request.
	error	This is the type of error that occurred. The following are the possible values: • Invalidrequestid
	description	This is a more detailed description of the error. The following are the possible values: • Invalidrequestid - The request id xxxx does not match any outstanding `<submit>` requests. xxxx is the value of the requestid attribute.
queue.query.done		This event indicates the success of the query request.
	requestid	This is the ID of the `<query>` request.
	priority	This is the current priority of the request.
	positioninqueue	This is the current position of the request in the queue.
	invqwaittime	This is the expected wait for the queue in relationship to the request.
	totalsize	This is the total number of agents that

Event	Attributes	Description
		can be targetted for the request. (Only since URS 8.1.3)
	loginsize	This is the number of agents logged in that can be targetted for the request. (Only since URS 8.1.3)
error.queue.query		This indicates that an error occurred while trying to perform the `<query>` request.
	requestid	This is the ID associated with the request.
	error	This is the type of error that occurred.
	description	This is a more detailed description of the error
queue.default.done		This event indicates the success of the request and that a default target has been selected.
	requestid	This is the ID associated with the request.
	defaultselected	This is the default configured target address.
queue.default.requestid		This event provides the application with request ID for the given request that was invoked.
	requestid	This is the ID associated with the request from the orchestration application or the resource.
error.queue.default		This indicates that an error occurred while trying to perform the `<default>` request.
	requestid	This is the ID associated with the request.
	error	This is the type of error that occurred: • timeout • invalidattribute

Event	Attributes	Description
		• abandoned • unknown • invalidstate.state (null, ringing, hold, transferring, treating, routed) • badtranslation • remote
	description	This is a more detailed description of the error. The following are the possible values: • timeout - the interaction was not redirected to the default target in a timely manner. • invalidattribute - The attribute yyy:xxx has an invalid value (zzz) or is not allowed under the conditions of the request. yyy is the name of the element associated with the attribute. xxx is the name of the attribute. zzz is the value of the attribute. • abandoned - The customer has abandoned the interaction. • unknown - The cause of the failure is unknown. • invalidstate.state - The interaction is in an invalid state and cannot be redirected to the default target. • badtranslation - The destination address xxx could not be translated. xxx is the address of the default target. • remote - There was an error in the media server while trying to redirect the interaction to the default target.

The following are the queue asynchronous events:

Event	Attributes	Description
queue.cancelled		This event indicates that a target has been selected from another `<submit>` action request and that this `<submit>` action request has been cancelled.
	requestid	This is the submit request ID associated with the interaction.

Notes

Prior to version 8.1.200.27, the Interaction's age was ignored for Multimedia Interactions.