



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Genesys Mobile Services Developer's Guide

Custom Reporting

12/14/2025

Contents

- 1 Custom Reporting
 - 1.1 Basic Configuration for Real-Time and Historical Reporting Based on T-Server's UserEvent Mechanism

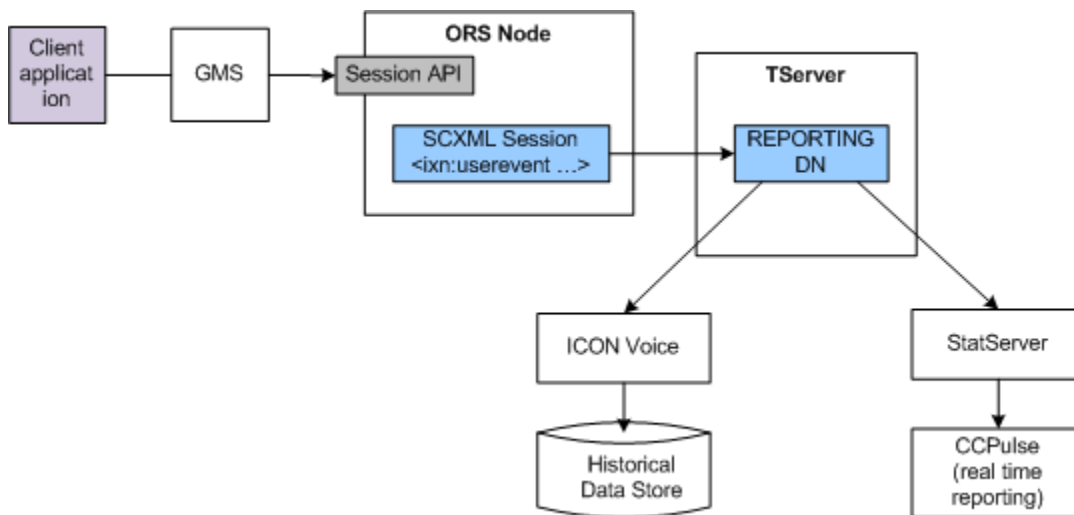
Custom Reporting

Basic Configuration for Real-Time and Historical Reporting Based on T-Server's UserEvent Mechanism

Prerequisites

1. ORS is connected to T-Server
2. StatServer is connected to the same T-Server (if you need real-time reporting)
3. Icon is connected to T-Server and configured to store user events to G_CUSTOM_DATA_P table (if you need historical reporting)

Architecture



Configuration instructions

- Create a new DN of type **Extension**. The name of the DN is not important, but it is used inside SCXML scripts so should be meaningful and recognizable.
Example: Sip_Switch -> DN -> REPORTING
- Make sure Icon and StatServer are connected to the T-Server that is servicing the switch specified in step 1.
Example: Sip_Server.
- In Icon configuration in Configuration Manager, add the 'custom-states' section under Options and create the **GlobalData** option there. List attached data fields you want to capture preceded with data

type.
Example:

```
custom-states/GlobalData=char,gms_SessionId, char,gms_SessionEventSeq, char,gms_ServiceName,  
char,gms_UserId, char,gms_externalId,  
char,gms_ServiceStartAt, char,gms_WaitingForAgent,  
char,gms_AgentAvailable, char,gms_UserConnected,  
char,gms_AgentConnected, char,gms_IxnCompleted,  
char,gms_ServiceStoppedAt
```

- Start Icon and StatServer (if not already started) and use logs to verify they registered on REPORTING DN.
- Add the following block of code to the beginning of your SCXML flow. This code will setup the `_data.userevent_udata_to_send` variable to store all significant state changes you want to capture from the point of view of reporting. Names should match Icon's GlobalData configuration option and StatServer/CCPulse reporting and statistics templates.
Example:

```
<datamodel>  
  
</datamodel>  
<script>  
  _data.userevent_udata_to_send = {  
    'gms_SessionId':_sessionid,  
    'gms_SessionEventSeq':0,  
    'gms_ServiceName':'your service name here',  
    'gms_UserId':,  
    'gms_externalId':,  
    // service state change timestamps  
    'gms_ServiceStartAt': ,  
    'gms_WaitingForAgent':,  
    'gms_AgentAvailable':,  
    'gms_UserConnected':,  
    'gms_AgentConnected':,  
    'gms_IxnCompleted':,  
    'gms_ServiceStoppedAt':  
  };  
</script>
```

- Add following block of code into your SCXML flow where significant state change is happening:

```
<script>  
  _data.userevent_udata_to_send.gms_WaitingForAgent = new Date().getTime().toString();  
  _data.userevent_udata_to_send.gms_SessionEventSeq =  
_data.userevent_udata_to_send.gms_SessionEventSeq + 1;  
</script>  
<ixn:userevent requestid="_data.userevent_reqid" resource="({'switch':'SIP_Switch',  
'dn':'REPORTING'})"  
  udata="_data.userevent_udata_to_send"/>
```

Example:

```
<state id="waitForAgent">  
  <onentry>  
    <script>  
      _data.userevent_udata_to_send.gms_WaitingForAgent = new  
Date().getTime().toString();  
      _data.userevent_udata_to_send.gms_SessionEventSeq =  
_data.userevent_udata_to_send.gms_SessionEventSeq + 1;  
    </script>  
  </onentry>  
</state>
```

```
    </script>
    <ixn:userevent requestid="_data.userevent_reqid"
resource="{ 'switch': 'SIP_Switch', 'dn': 'REPORTING' }"
              udata="_data.userevent_udata_to_send"/>

    <queue:submit route="false" timeout="_data.queueSubmitTimeout">
      <queue:targets type="agentgroup">
        <queue:target name="_data.defaultAgentGroup"/>
      </queue:targets>
    </queue:submit>
  </onentry>

  <transition event="queue.submit.done" target="agentAvailable"/>
  <transition event="error.queue.submit" target="error">
  </transition>
  <transition event="service.ttl.expired" target="error">
  </transition>
</state>
```

Verifying reporting data

- Run your scenario by triggering Genesys Mobile Services and Orchestration Server (ORS) APIs directly.
- Make sure user events are being delivered to StatServer and Icon applications by checking T-Server logs. You should see something like this:

```
00:34:20.757 Int 04543 Interaction message "RequestDistributeUserEvent" received from 516
("OrchestrationServer")
-- Absent ThisDN, REPORTING was used
@00:34:20.7570 [0] 8.1.000.62 send_to_client: message EventACK
  AttributeEventSequenceNumber      00000000000000ef8
  AttributeCustomerID                'Environment'
  AttributeTimeinUsecs              757000
  AttributeTimeinSecs                1348817660 (00:34:20)
  AttributeReferenceID               431
  AttributeThisDN                    'REPORTING'
  AttributeUserEvent                  RequestDistributeUserEvent
00:34:20.757 Trc 04542 EventACK sent to [516] (00000003 OrchestrationServer
192.168.27.50:40727)
@00:34:20.7570 [0] 8.1.000.62 distribute_user_event: message EventUserEvent
  AttributeEventSequenceNumber      00000000000000ef9
  AttributeCustomerID                'Environment'
  AttributeTimeinUsecs              757000
  AttributeTimeinSecs                1348817660 (00:34:20)
  AttributeUserEvent                  EventUserEvent
  AttributeUserData                  [347] 00 0c 00 00..
    'gms_AgentAvailable'             '1348817660755'
    'gms_AgentConnected'
    'gms_IxnCompleted'
    'gms_ServiceName'                'inbound-delay'
    'gms_ServiceStartAt'             '1348817660553'
    'gms_ServiceStoppedAt'
    'gms_SessionEventSeq'            3
    'gms_SessionId'                  '65UA6ISSJH76R340BNDQ2DG0DG000036'
    'gms_UserConnected'
    'gms_UserId'
    'gms_WaitingForAgent'            '1348817660744'
    'gms_externalId'
  AttributeANI                       '777'
  AttributeDNIS                       '333'
  AttributeReferenceID               431
```

```
AttributeThisDN      'REPORTING'
00:34:20.758 Trc 04542 EventUserEvent sent to [508] (0000000c Icon_Voice 192.168.27.50:42678)
00:34:20.758 Trc 04542 EventUserEvent sent to [588] (00000004 Stat_Server
192.168.27.50:40728)
00:34:20.758 Trc 04542 EventUserEvent sent to [592] (00000005 Universal_Routing_Server
192.168.27.50:40744)
```

- Check your Icon log and G_CUSTOM_DATA_P table and make sure data is recorded properly.
Examples:

Icon log:

```
00:39:19.569 Int 04543 Interaction message "EventUserEvent" received from 65200
("SIP_Server@REPORTING")
00:39:19.751 Int 04543 Interaction message "EventUserEvent" received from 65200
("SIP_Server@REPORTING")
00:39:19.766 Int 04543 Interaction message "EventUserEvent" received from 65200
("SIP_Server@REPORTING")
00:39:19.987 Trc 25016 Persistent Queue GUD: transaction 10929 is committed. 5 records
written into the queue
00:39:19.987 Trc 25003 Database queue [GUD]: persistent queue transaction 10929 is being
processed.
00:39:20.001 Trc 25004 Database queue [GUD]: persistent queue transaction 10929 is
processed, committed and removed. 5 records are written.
```

Icon's G_CUSTOM_DATA_P table:

```
select * from dbo.G_CUSTOM_DATA_P

8      0      830      REPORTING      0      101      1      2012-09-28
07:43:09.443      1348818189      4496060      65UA6ISSJH76R340BNDQ2DG0DG000038
1      inbound-delay      1348818189441
9      0      830      REPORTING      0      101      1      2012-09-28
07:43:09.590      1348818189      4496060      65UA6ISSJH76R340BNDQ2DG0DG000038
2      inbound-delay      1348818189441      1348818189590
10     0      830      REPORTING      0      101      1      2012-09-28
07:43:09.600      1348818189      4496060      65UA6ISSJH76R340BNDQ2DG0DG000038
3      inbound-delay      1348818189441      1348818189590
1348818189596
```

- Start CCPulse and create a reporting template for monitoring REPORTING DN.

Congratulations: you are done!