Softdial Scripter Integration Guide

Introduction

Softdial Scripter is a general-purpose call center scripting tool that enables scripting of business processes within the call center.

Its primary use is as an engine to deliver web-based agent desktop scripting. Softdial Scripter can be delivered as a standalone product, to be integrated with a third-party call center communications platform. This document provides details of the interface for third-party applications.

Interface Technology

Softdial Scripter requires integration with various aspects of a call center's operation – telephony, database and agent desktop are 3 contact points with specific requirements. To simplify integration, Sytel has developed a service that delivers a simple web services API to act as a concentration point.

The Web services API is a simple http based API offering URL invocation for passing events to the scripter server (such as call X has connected to agent Y) and provides an event polling interface via http for events that the scripting environment needs to deliver (such as agent Y has completed processing the call, has dispositioned the call and is now available again).

The URL invocation interface returns an XML document in a simple and standard form.

The main reason we are offering a standard http interface is to provide a lowest-common-denominator means to integrate with scripter, and to provide the means to test partial integrations. Since the http interface can be driven by typing an extended URL into a browser this makes exploration of the capabilities easy. Most programming languages have constructs for managing http requests and responses.

A secondary reason is that Softdial Scripter is designed to be hosted. Hosted applications must offer up standards-based interfaces that can be routed over the public internet. This is the reason why the http API is half-duplex as per normal http request/response patterns.

XML Schemas

Two main schemas are employed in the API.

The first is the return document. The return document is sent in response to every valid http request received by the interface. In the event of a URL or parameter error an http error code will be returned. In the event that a properly-framed request is made, a 200 OK response will always be sent back to indicate that the message was properly-framed. The body of the response will be an XML document containing the following elements –

`<id>` - the unique identifier for the communication session between the third-party.

`<acc>` - (Accepted) a Boolean (string true/false) stating whether the message has been accepted for processing.

`<re>` - (Reason)If not accepted, reason contains a plain text reason why the message was not accepted. For example, if a 'connect call to agent' request is sent for an agent that is on another call the reason will explain this.

`<ev>` - (Event) A numeric event code to enable consistent processing of http responses. A normal response to a request from a third-party application will contain the value 0. An error response will have the value -1. vents raised by scripter to advise the third-party call center application will have positive integer values as documented.

`<cu>` - (Custom) The custom section may contain data pertinent to the request or event raised by scripter.

A sample positive acknowledgement response is shown below.

```
< ScripterEvent xmlns="sytelco.com/SCC/SSB">
  <id>abcd1234</id>
  <acc>true</acc>
  <re />
  <ev>0</ev>
  <cu />
</ScripterEvent>
```

The second schema is the data transfer schema. When advising scripter of an agent connection it is also necessary to provide line-of-business data as part of the event. The data transfer schema is an XML representation of a dictionary of keyword-value pairs representing field name and field value, for populating the script.

Web service interface details

The scripter integration service by default listens on port 91 for http. Since scripter by default listens on port 80, if the integration service is deployed on the same host it will need to listen on a different port. If a single port 80 interface is desired, setting up Apache as a proxy for both scripter and the integration service is a simple exercise.

Commands

Session

The session command registers a new interface session for the integrator. The session command has parameters for authentication. Once a session has been registered it can be used to handle other commands and pick up events.

An integrator would normally use a single session to manage a server-side integration with scripter. This does not have to be the case. In certain circumstances it may prove beneficial for the integrator to implement certain of the commands against a client application, in which case a session may be established for each client connection.

Parameters to the session command
id – the session identifier
td – a tenant descriptor for an authenticated user.
an – a user name for an authenticated user.
pw – the password for an authenticated user.

Example

http://localhost:91/ScripterBridge/session?id=garry&td=default&an=Garry&pw=Garry

Note – the user for a server-side integration should be registered in the Softdial namespace as a super-user.

Close

The close command closes a scripter integration session and removes any and all resources associated with the session.

Parameters to the session command
id – the session identifier

Example

http://localhost:91/ScripterBridge/close?id=garry

Tenant Start

The Tenant Start command is optional and should only be implemented if the integrator provides support for multitenancy. In this case each command requires a tenant id (td) parameter. If the integrator does not support multitenancy this command can be ignored.

Parameters to the tenant start command
id – the session identifier
td – the tenant descriptor for the new tenant

Example

http://localhost:91/ScripterBridge/tenantstart?id=garry&td=newtenant

Tenant Stop

As with tenant start, tenant stop only needs to be implemented if the integrator provides support for multitenancy.

Parameters to the tenant start command
id – the session identifier
td – the tenant descriptor for the new tenant

Example

http://localhost:91/ScripterBridge/tenantstop?id=garry&td=newtenant

Note – sending the tenantstop command will bring all campaigns for that tenant to a halt, as soon as agents have finished their script sessions.

Campaign Start

The campaign start command indicates to Scripter that a campaign has started, and agents can therefore log on to the campaign. The campaign may be inbound or outbound. The set of events exposed via the integration API apply to both types of campaign.

Parameters to the campaign start command
id – the session identifier
td – the tenant descriptor for the new tenant. Optional, omit if no multitenancy
cn – the campaign name

Example

http://localhost:91/ScripterBridge/campaignstart?id=garry&cn=test

Campaign Stop

The campaign stop command indicates to Scripter that a campaign has stopped.

Parameters to the campaign stop command
id – the session identifier
td – the tenant descriptor for the new tenant. Optional, omit if no multitenancy
cn – the campaign name

Example

http://localhost:91/ScripterBridge/campaignstop?id=garry&cn=test

Note – this command should be sent to indicate that the campaign has stopped. If the command is sent whilst agents are logged on and processing calls, the agent sessions will be terminated.

Station logged in

The station logged in command signals to Scripter that an agent has completed ACD login. The login process starts when the user, or the integrator's application invokes a scripter logon URL, indicating that a particular agent wants to log on to a particular campaign. Scripter relays this back to the integrator via an event which would normally trigger station nail-up for ACD operation. The station logged in command is sent by the integrator to indicate that nail-up or ACD login is complete. Once this command is received by the scripter integration service, the agent is logged in and made ready in scripter.

Parameters to the station login command are
id – the session identifier
td – the tenant descriptor for the new tenant. Optional, omit if no multitenancy
cn – the campaign name
an – the agent name

Example

http://localhost:91/ScripterBridge/stationlogin?id=garry&cn=test&an=1

Note – agent names must be unique within a tenant (or within the system as a whole if multitenancy is not used)

Station logged out

The station logged in command signals to Scripter that an agent has logged out of the system.

Parameters to the station login command are
id – the session identifier
td – the tenant descriptor for the new tenant. Optional, omit if no multitenancy
cn – the campaign name
an – the agent name

Example

http://localhost:91/ScripterBridge/stationlogout?id=garry&cn=test&an=1

Note – if the agent is logged in and processing a call, this will cause the script to end and the agent to be logged out. This command should only be used to deal with agent kill scenarios or if the agent station can no longer be reached.

Agent connected

The agent connected command indicates to scripter that a call has been connected to an agent and specifies some line-of-business data for the call to be used in scripts.

Parameters to the agent connected command are
id – the session identifier
td – the tenant descriptor for the new tenant. Optional, omit if no multitenancy
cn – the campaign name
an – the agent name
dt – the line-of-business data to be managed by scripter

Example

http://localhost:91/ScripterBridge/agentconnect?id=garry&cn=test&an=1&dt=foo

Note – in the example the dt parameter is a substitute for the data parameter format string, which will usually be a long xml string.

Agent hung up

The agent hung up command signals to scripter that either the agent has dropped the customer call, or that the customer has hung up, or if the agent is not an off-hook agent, the agent has hung up.

Parameters to the agent hung up command are
id – the session identifier
td – the tenant descriptor for the new tenant. Optional, omit if no multitenancy
cn – the campaign name
an – the agent name

Example

http://localhost:91/ScripterBridge/agenthangup?id=garry&cn=test&an=1

Get Events

The Get Events command should be called by the 3rd-party application every few seconds to monitor for events reported from scripter. This first version of the Scripter integration service supports only a single event although more are planned.

Parameters to the Get Events command are
id – the session identifier

Example

http://localhost:91/ScripterBridge/getevent?id=garry

Note – in the event that there are no events the <ev> element of the response will have the value -1. All events generated by scripter have a positive number value for the <ev> element.

Event parameters

A typical response to the Get Events command follows the standard xml document specification with some extra embedded data

Following is an example of the 'agent requested logon' event

```
<ScripterEvent xmlns="sytelco.com/SCC/SSB">
  <id>garry</id>
  <acc>true</acc>
  <re />
  <ev>1</ev>
  <cu>
      <ScripterLogonEvent xmlns="sytelco.com/SCC/SSB"><td>default</td>
        <cn>test</cn>
        <an>1</an>
        <ae>1</ae>
      </ScripterLogonEvent>
  </cu>
</ScripterEvent>
```

Note the custom <cu> element contains an embedded xml document as its text. In the case of the logon request the agent properties are embedded.