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 Administrator Extension Deployment Guide

Logging In Remotely

4/13/2025

Contents

Logging In Remotely

Genesys Administrator Extension supports three types of remote logins, as follows:

- Customized Login Page
- Whitelisted Hosts
- Login API

The following three parameters specify to where the user is directed in the case of a successful or unsuccessful remote login, and logout. While not mandatory, you might want to define a list of trusted URLs to which these three variables can be set.

- `login_success_url`—Set this to the URL for the GAX login page. If this value is not set, the page is redirected to itself and the URL is appended with `#success`.
- `login_failure_url`—Set this to a URL to which the user will be directed if the supplied credentials are invalid. If this value is not set, the page is redirected to itself and the URL is appended with `#failure`.
- `logout_url`—Set this to a URL to which the user will be directed after logging out of GAX. If this value is not set, the user is redirected to the initial login screen and the URL is appended with `#logout`.

Customized Login Page

Users can log in to GAX using a customized Login Page that is located on another website (for example, a corporate portal page). In this scenario, the company network can pass the user's credentials to GAX, and GAX automatically logs in the user via a background process so that the user bypasses the login screen. In addition, a logout URL can be set so the user returns to the company portal page after logging out of GAX.

To use this feature, the customized login page must submit a form to the GAX login page. The following is an example:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<script>
"use strict";
function logonGax(Form)
{
  var xhr = new XMLHttpRequest();
  xhr.open (Form.method, Form.action, true);
  xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  xhr.setRequestHeader("Access-Control-Allow-Origin", "*");
  xhr.send (new FormData (Form));
}
</script>
</head>
<body>
```

```
<form id="logon" action="http://localhost:8080/gax/api/session/login" method="post"
onsubmit="logonGax(this)">
<p>UserID:<input type="text" name="username"></p>
<p>Password:<input type="password" name="password"></p>
<input type="hidden" name="newPassword" value="">
<input type="hidden" name="newPasswordConfirm" value=""></p>
<input type="hidden" name="login_success_url" value="https://localhost:8080/
gax/?#/!/view:com.cm.home/1">
<input type="hidden" name="logout_url" value="https://www.google.com">
<input type="hidden" name="login_failure_url" value="https://www.google.com">
<p><input type="submit" value="Submit"></p>
</form>
</body>
</html>
```

Whitelisted Hosts

If you have not done so already, define a list of one of more trusted hosts, called whitelisted hosts, to which these three variables can be set. Then in the **[security]** section, set the following options:

- **host_whitelist_enabled**=`true`
- **host_whitelist**=<semicolon-separated list of hosts specified by `login_success_url`, `login_failure_url`, and `logout_url`>

See more details about these two options here.

> ### Important
>
> The `host_whitelist` option is not meant for blocking/permitting IP addresses. It is only used for defining login, logout, and failure redirect URLs of GAX and not for individual IP addresses. If you want to restrict access to GAX URLs, Do it at System or Network level. Please consult your System or Network Administrator for the same.

Example

Using the same values used in the example above, assume the list of trusted hosts has been defined and assigned to the three variables, as follows:

- `login_success_url=https://localhost:8080/gax/?#/!/view:com.cm.home/1`
- `login_failure_url=https://www.google.com`
- `logout_url=https://www.google.com`

The option assignments would then be as follows:

```
[security]
host_whitelist_enabled=true
host_whitelist=https://localhost:8080/gax/?#/!/view:com.cm.home/
```

```
1;https://www.google.com;https://www.google.com
```

Login API

You can use a login API, such as:

Function:	Login (cross domain)
URL:	`/session/login`
Method:	POST
Content type:	`application/x-www-form-urlencoded`
Request Body:	`{ "username":"default", "password":"password", "newPassword": "password2", "newPasswordConfirm": "password2" "login_success_url": "http://xyz.com/success.html" "login_failure_url": "http://xyz.com/failure.html" "logout_url": "http://xyz.com/logout.html" }`

When calling the function, precede the URL with **http://<gaxserver>:<port>/api**.