broadsoft

broadsoft broadworks

Application Server API

Developer's Guide

Release 21.0

Document Version 1

9737 Washingtonian Boulevard, Suite 350
Gaithersburg, MD 20878
Tel +1 301.977.9440

WWW.BROADSOFT.COM

BroadSoft® Guide

Copyright Notice

Trademarks

Any product names mentioned in this document may be trademarks or registered trademarks of BroadSoft or their respective companies and are hereby acknowledged.

This document is printed in the United States of America.

Document Revision History

Release	Version	Reason for Change	Date	Author
14.0	1	Updated document for re-branding.	February 1, 2006	Roberta Boyle
14.0	1	Made Release 14 updates.	June 27, 2006	Michel Tougas
14.0	1	Edited document.	August 7, 2006	Patricia Renaud
15.0	1	Updated document for Releases 14.sp1 through 14.sp6 and Release 15.0.	July 7, 2008	Patricia Renaud
16.0	1	Updated document for Release 16.0.	June 11, 2009	Yves Racine
16.0	1	Edited changes and published document.	August 26, 2009	Andrea Fitzwilliam
17.0	1	Updated document for Release 17.0	February 21, 2010	Yves Racine
17.0	1	Edited and published document.	March 24, 2010	Margot Hovey-Ritter
18	1	Updated references to the *BroadWorks OSS Developer's Guide*, which were replaced by the *BroadWorks Network Server Provisioning Interface Specification*.	September 27, 2011	Steve Davis
18	1	Edited changes and published document.	October 5, 2011	Jessica Boyle
19	1	Updated document for Release 19.0.	October 17, 2012	Michael Boyle
19	1	Edited changes and published document.	October 24, 2012	Patricia Renaud
20	1	Updated document for Release 20.0.	October 18, 2013	Michael Boyle
20	1	Edited changes and published document.	October 18, 2013	Patricia Renaud
21	1	Updated document for Release 21.0.	September 25, 2014	Michael Boyle
21	1	Edited changes and published document.	December 19, 2014	Patricia Renaud

Table of Contents

Table of Figures

1 Summary of Changes

This section describes the changes to this document for each release and document version.

1.1 Changes for Release 21.0, Document Version 1

There were no changes to this document for Release 21.0.

1.2 Changes for Release 20.0, Document Version 1

There were no changes to this document for Release 20.0.

1.3 Changes for Release 19.0, Document Version 1

There were no changes to this document for Release 19.0.

1.4 Changes for Release 18.0, Document Version 1

Update the references to the BroadWorks OSS Developer's Guide – replaced by the BroadWorks Network Server Provisioning Interface Specification.

1.5 Changes for Release 17.0, Document Version 1

There were no changes to this document for Release 17.0.

1.6 Changes for Release 16.0, Document Version 1

In this version of the document, the Application Server portal API no longer exists and is replaced by the corresponding OCI commands.

1.7 Changes for Release 15.0, Document Version 1

There were no changes to this document for Release 15.0.

1.8 Changes for Releases 14.sp1 through 14.sp6, Document Version 1

There were no changes to this document for Releases 14.sp1 through 14.sp6.

2 Purpose

This document describes how to use the portal application programming interface (API) for the Application Server. It is intended as a programming guide for service providers who want to "front-end" the Application Server through their portal Web Server. This gives service providers a mechanism to use that seamlessly integrates the CommPilot Call Manager services with their existing portal services. The goal of the portal API is to allow a service provider to offer an end-user experience that does not require the end user to log in multiple times.

The features offered by the portal API include the ability to:

- Locate the Application Server hosting a particular user's services and CommPilot Call Manager properties.
- Redirect the user's browser to the hosting Application Server.
- Automatically log the user into the hosting Application Server.
- Locate the primary Application Server in a replicated cluster (used for administrator login).

This document is intended for system integration developers. There are three documents that are used in conjunction with this document: *Application Server Provisioning Interface Specification Guide* [4], *Network Server Portal API Specification* [1], and *BroadWorks Network Server Provisioning Interface Specification* [3].

3 Overview

The portal API is a set of APIs for two different components. There is one for the Network Server and two for the Application Server.

The location API of the Network Server allows the portal to resolve the location of the user, that is, the Application Server hosting the user's services and CommPilot Call Manager properties. This is an XML-over-HTTP interface. For information on commands available in this XML API, see the *Network Server Portal API Specification*.

The login API of the Application Server allows the portal to send the user's browser to the appropriate Application Server *Login* page, and optionally to auto-log the user into the system. This is a formatted Hypertext Transfer Protocol Uniform Resource Locator (HTTP URL) with appropriate *common gateway interface (CGI)* parameters.

The open client interface (OCI) API offered by the Application Server allows the portal to resolve the location of the primary Application Server in a replicated cluster, that is, the Application Server that should be used for configuration changes by administrators. In addition, this API offers the capability to make logging in more secure and transparent, through the login token feature.

Figure 1 summarizes the interfaces and interactions for the portal API. The following sections describe how to use the API.

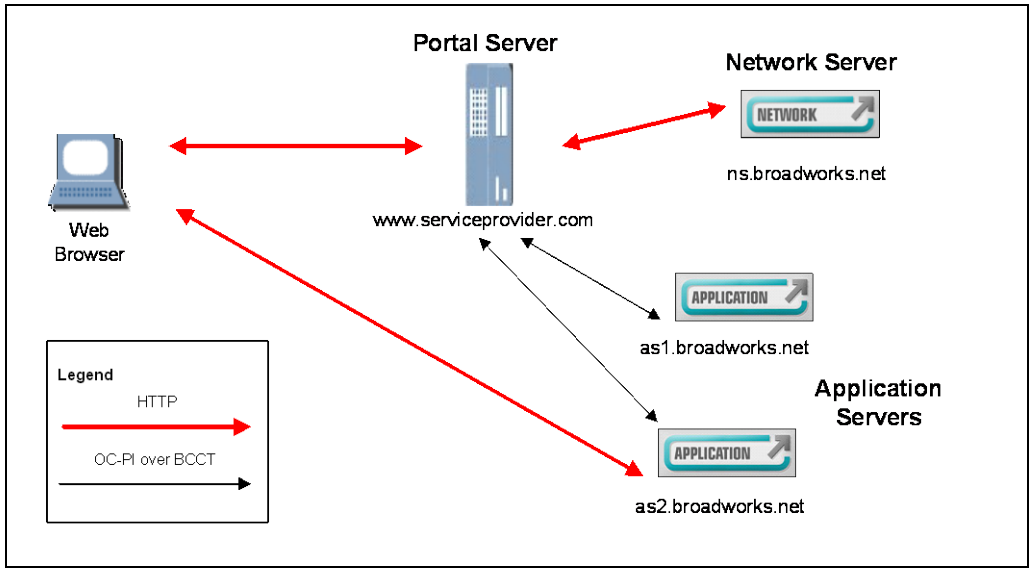

Figure 1 Portal Application Programming Interface Connections

There are three scenarios to be considered:

- End-user login to an Application Server
- Administrator login to a non-replicated Application Server
- Administrator login to a replicated Application Server

4 End-user Login

The steps involved in using the location and login APIs for an end-user login are as follows:

1) The user logs into the portal server and navigates the web pages. The user clicks on a link with the intention of going to the CommPilot Call Manager.
2) The portal server uses the location API on the Network Server to resolve the hostname of the appropriate Application Server for the user.
3) The Network Server returns the address of the end-user's hosting Application Server.
4) The portal server uses Open Client Interface-Provisioning (OCI-P) command ExternalAuthenticationAuthorizeTokenRequest on the identified Application Server to prepare a login token. This step is necessary to hide the user ID and password from the user (on the Application Server). If this is not a concern, this step can be skipped.
5) The Application Server returns a successful response.
6) The portal server redirects the user's browser to the Application Server with appropriate credentials (prepared token or user ID/password) to automatically log the user in. As part of this step, the portal server can optionally specify a redirect URL for the user when the web session expires. Usually, the user is presented with the Application Server *Login* page when the browser session expires.
7) The end-user's web browser accesses the login servlet on the Application Server with the provided parameters.

The same sequence applies when the Application Server is stand-alone (non-replicated) as shown in *Figure 2* or replicated (clustered) as shown in *Figure 3*. In either case, the location API request (steps 2 and 3) returns the address of the exact Application Server hosting the user. It is important to note that in some networks, a Network Server may not be deployed, as shown in *Figure 4*. In this case, the portal server must perform its own location resolution (steps 2 and 3) to find the appropriate Application Server, and then continue to use the Application Server login API (as described in step 4).

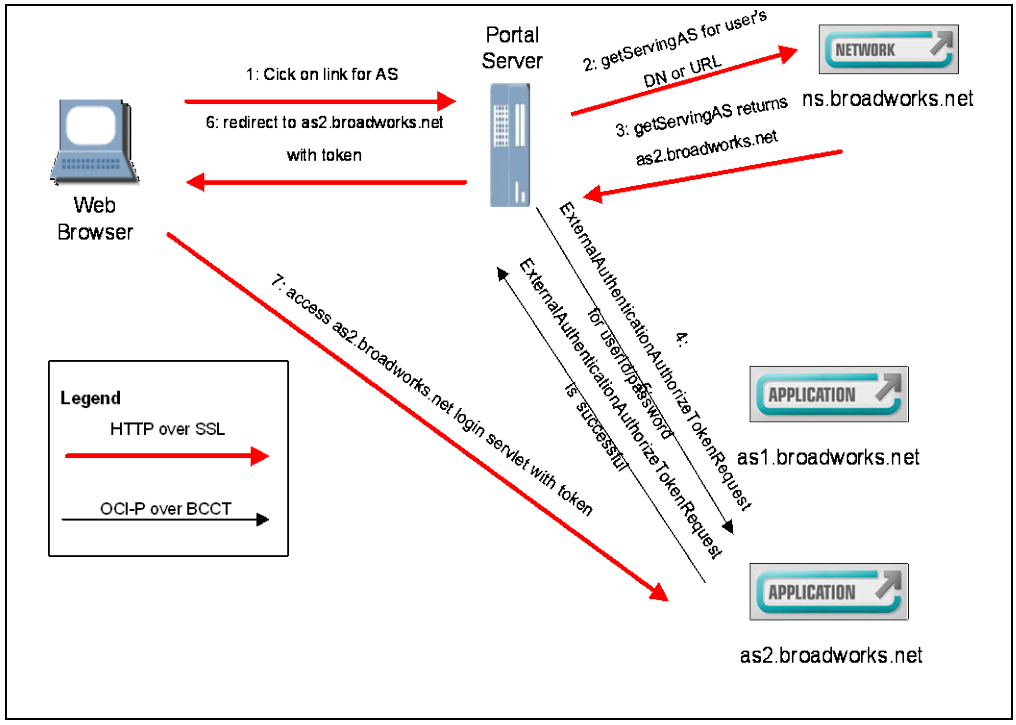

Figure 2 End-user Login (Non-replicated Scenario)

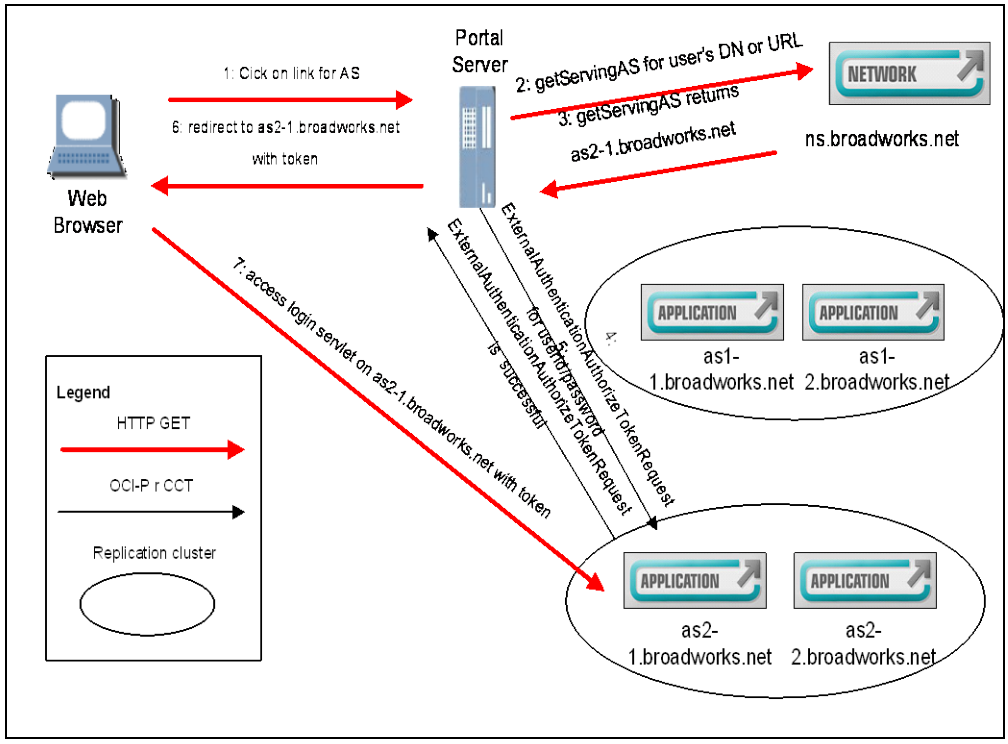

Figure 3 End-user Login (Replicated Scenario)

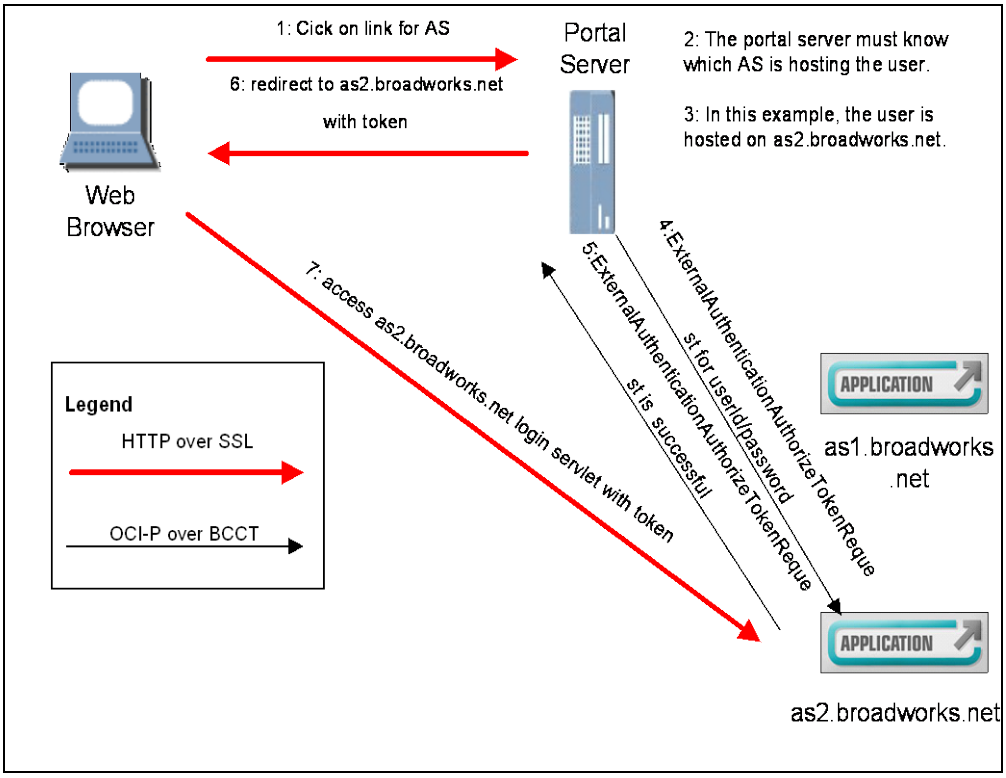

Figure 4 End-user Login (Scenario without a Network Server)

The user has the option of being directed to the Application Server and presented with the standard Application Server *Login* page, or choosing to log in to the Application Server automatically.

4.1 Portal Side End-user Profile

To use the API, the portal server must maintain a profile of attributes associated with each user that can access the CommPilot Call Manager. These attributes, shown in *Table 1*, are used as parameters to each of the APIs.

Attribute Name	Attribute Type	API	Description
user address	String	Location	In a redundant scenario or in a non-redundant scenario with a Network Server, this is the fully qualified E.164 (for example, 11 digits in North America) telephone number, or a fully qualified SIP URI that is associated with the user on the Application Server. In a non-redundant scenario without a Network Server, the hosting Application Server address cannot be obtained from the Network Server. Therefore, the portal-side user data should contain the actual address of the user's hosting Application Server. Obviously, the location API look up is skipped and the hosting address is used directly in the subsequent steps of the login process.
user ID	String	Login/Location	BroadWorks Application Server login ID associated with the user.
user password	String	Login/Location	BroadWorks Application Server password associated with the user's login ID. If external authentication password rules are enabled on the Application Server, and if the user has a blank password, then the password attribute value for the location API is an empty string. Blank passwords are not supported directly in the login API. A login token must be prepared using the location API, after which the user can log in through the login API using that token.

Table 1: End-user Profile Attributes

4.2 Locate User

The procedure for using the location API follows the same general steps as those used for the Network Server provisioning Operations Support System (OSS) API. This interface uses a typical XML-over-HTTP communication model. All requests are packaged into an HTTP request (with URL parameters) by the client, and then passed to the Network Server through a simple HTTP GET request. The Network Server processes the request, generates an appropriate response document, and hands it back to the caller.

For complete details on the location API XML commands, see the *Network Server Portal API Specification* [1].

4.2.1 Build Request

Before the portal server can use the location API, it must build a valid HTTP GET request with an appropriate set of URL parameters. The portal server must invoke the "GetServingAS" servlet and must provide a means by which the user to be located is specified.

```
HTTP GET
http://mtlns03/servlet/GetServingAS?dn=%2B15146987500
```

As part of the request URL, the portal server must include a URL parameter that identifies the user to be located. The following example uses the directory number as the identifying parameter:

```
HTTP GET
http://mtlns03/servlet/GetServingAS?dn=15146971001
```

Optionally, a URL could be used instead:

```
HTTP GET
http://mtlns03/servlet/GetServingAS?url=dude%40serviceprovider.com
```

If the Application Server is set up in a dual-homed configuration, with the web application available from both the public (access) and private (signaling) networks, the "private" URL parameter can be used to indicate the type of address to be returned. By default, public access addresses are returned.

If the Application Server has an external Web Server (see Release 10 External Web Server Support functionality), then the public address is the external Web Server address, while the private address is the address of the collocated Web Server on the Application Server host.

4.2.2 Process Request

Once the portal server has built the HTTP request, it must send it to the Network Server for processing. The Network Server requires all location API requests, such as the GetServingAS request, to originate from a pre-authorized portal server. This means that the portal server's address used to send a request to the Network Server must first be provisioned into the portal API network access list control on the Network Server, under CLI level *NS_CLI/System/NetworkAccessLists/PortalAPI*.

4.2.3 Decode Successful Response Document

The HTTP GET response is a HTTP 200 OK message containing a stringed XML-response document. The document has attributes and values set to indicate the results of the request. A successful request is similar to the following:

```
HTTP 200 OK text/html
<?xml version="1.0" encoding="UTF-8"?>

<com.broadsoft.protocols.nsportal.PortalRequest
 dn="15146971001"
 isPrivate="false"
 url=""
>
  <addressArray>
    <string value="as1.broadworks.net"></string>
  </addressArray>
</com.broadsoft.protocols.nsportal.PortalRequest>
```

A portal server can decode this response and determine the success of the request and the corresponding results. Lines shown in bold are attributes added by the Network Server to indicate the response of processing the original request. In a successful response, the root element is com.broadsoft.protocols.nsportal.PortalRequest. After the portal server has determined that the request succeeded, it can decode the response document further to determine the resulting value of the Application Server. This value is found in the addressArray element that has been added to the com.broadsoft.protocols.nsportal.PortalRequest element block.

The addressArray has one string-type element set to the value of "as1.broadworks.net". This is the (public) address of the Application Server hosting the user's CommPilot Call Manager services. For certain configurations (for example, cluster or multi-path), there can be more than one string element returned. The portal server should always try to connect to the first address in the list, and try subsequent addresses only if a connection to the first address cannot be established.

4.2.4 Decode Failure Response Document

It is possible for a request to fail. This could happen if the directory number (DN) or URL passed in the request does not match a provisioned user on the Network Server. The portal server should decode the response to determine if the request succeeded. A failed request would be similar to the following:

```
HTTP 200 OK text/html
<?xml version="1.0" encoding="UTF-8"?>

<com.broadsoft.protocols.nsportal.Error
  detail="DN is unassigned or cannot obtain public address for DN or
host may be OffLine "
  id="0"
  summary="Cannot complete request"
>
</com.broadsoft.protocols.nsportal.Error>
```

The Network Server returns the com.broadsoft.protocols.nsportal.Error root element, indicating that the request failed. Within this element, the "summary" attribute gives an error overview and the "detail" attribute explains why the request failed.

Receiving an error such as this usually indicates a provisioning problem on the Application Server or on the portal server. How the portal server handles this error depends on the systems integrator.

4.3 Prepare Login Token

If the end user's user ID and password on the Application Server must be hidden from the user, the location API on the Application Server must be used to prepare a login, before redirecting the user to the address identified in the preceding step.

To prepare a login token, the portal server must use the ExternalAuthenticationAuthorizeTokenRequest open client interface (OCI) command available in the location API on the Application Server, as shown in *Figure 5*.

For complete details on the OCI API, see the *Application Server Provisioning Interface Specification Guide* [4].

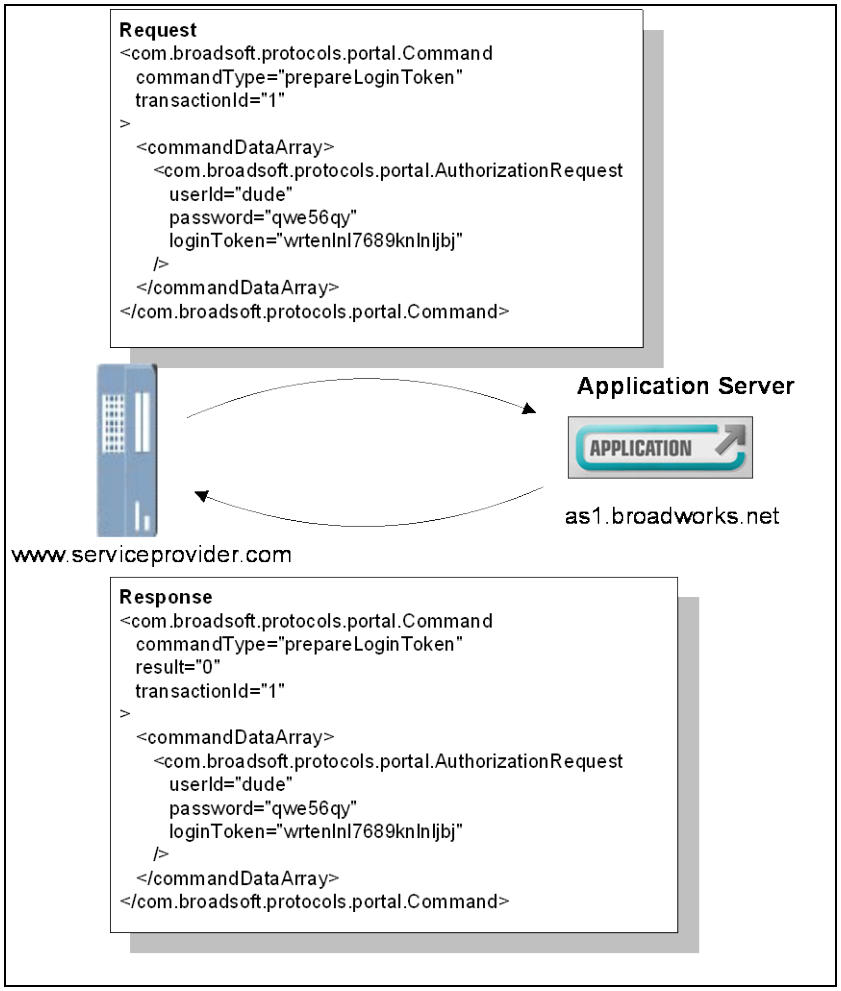

Figure 5 Location API on Application Server (ExternalAuthenticationAuthorizeTokenRequest OCI Command)

4.3.1 Build Request Document

Before the portal server can use the location API, it must build a valid XML request document with an appropriate set of commands. The portal server must build and send an ExternalAuthenticationAuthorizeTokenRequest OCI request, such as in the following example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<BroadsoftDocument protocol="OCI" xmlns="C"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <sessionId xmlns="">6907864BCFEEBCB98A6BB46EF07250DE</sessionId>
  <command xsi:type="ExternalAuthenticationAuthorizeTokenRequest"
xmlns="">
    <userId>dude</userId>
    <password>qwe56qy</password>
    <loginToken>wrtenlnl7689knlnljbj</loginToken>
  </command>
</BroadsoftDocument>
```

Inside the request, the portal server must include parameters that authenticate the user for which the token is being prepared. The user's user ID and password are passed in plain text. The third parameter is a unique string that is internally mapped to the user ID/password in preparation for a subsequent login attempt through the login API. The

loginToken is valid for 60 seconds, or until the login attempt is made, whichever comes first. It is the responsibility of the portal server to ensure the uniqueness of the login token.

4.3.2 Process Request

The Application Server processes the token request.

4.3.3 Decode Successful Response Document

If the Application Server processes the request successfully, it returns an OCI SuccessResponse. Any other response indicates an error. A successful response is similar to the following:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<BroadsoftDocument protocol="OCI" xmlns="C"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <sessionId xmlns="">6907864BCFEEBCB98A6BB46EF07250DE</sessionId>
  <command debugInfo="1:0" echo="" xsi:type="c:SuccessResponse"
xmlns:c="C" xmlns=""/>
</BroadsoftDocument>
```

4.3.4 Decode Failure Response Document

For an ExternalAuthenticationAuthorizeTokenRequest request, a failure is expected only if one of the required pieces of data is missing. All other failures (invalid user ID or password) are handled when the login attempt is made through the login API.

When there is a failure, the response is of type “ErrorResponse”. Elements “summary”, “summaryEnglish”, and “detail” explain the reason of the request failure.

4.4 Redirect User to Login Page

After the portal server has located the user and received a corresponding host name, it can then redirect the user to the appropriate Application Server with the login API. The login API is a CGI interface on the Application Server Web Server. To use it, the portal server must build an HTTP 302 redirect response.

The portal server must first build an appropriate URL for the redirect response. The URL must be of the following format:

```
http://ashost/servlet/Login
```

… where “ashost” is the hostname or IP address of the Application Server returned from the location API on the Network Server.

> **NOTE**: If “full” SSL support has been enabled on the Application Server, whether at installation or afterwards using the config-network utility, the above URL should specify the https:// protocol, especially if the portal web site is SSL-protected. Not specifying the appropriate protocol results in an unnecessary browser warning such as: “You are being redirected to a non-secure page”.

After the portal server has built the redirect URL, it can send the 302 redirect response back to the user, as shown in *Figure 6*.

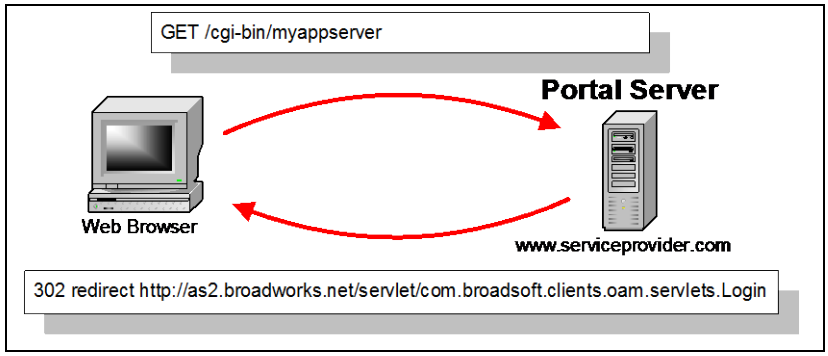

Figure 6 Interaction between End-user Browser and Portal Server

The end user's browser honors the 302 redirect by linking to the URL at:

http://as2.broadworks.net/servlet/Login

4.5 Automatic Login with Token

An advantage of the portal API is the ability to automatically log the user into the Application Server. This means that the user has previously been authenticated on the portal server and has a corresponding Application Server user ID and password in the user's portal server profile, and that a login token has been prepared as described in section *4.3 Prepare Login Token*. If these assumptions are correct, the portal server can send the login token as a *CGI* parameter in the redirect response:

```
http://ashost/servlet/Login?loginToken=preparedToken
```

Where ashost is the host name of the Application Server, and loginToken is the prepared token as shown in *Figure 7*.

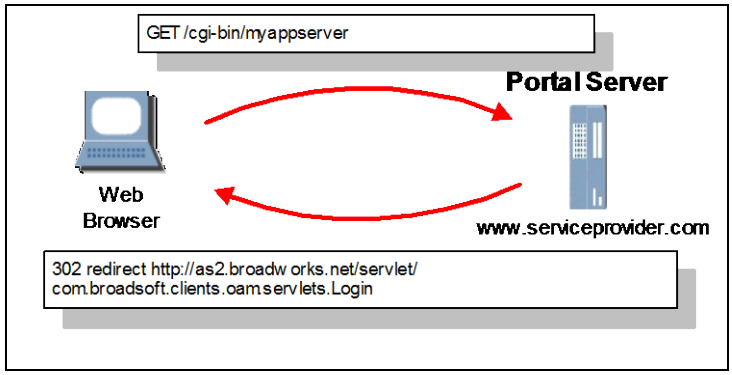

Figure 7 Interaction between End-user Browser and Portal Server with Automatic Login (Token)

4.6 Automatic Login with User ID and Password

If steps 4 and 5 were skipped, the end user can still be automatically logged into the Application Server. The portal server can send in the redirect response along with the corresponding user ID and password, as CGI parameters:[1]

```
http://ashost/servlet/Login
?UserID=userid&Password=password
```

Where "ashost" is the host name of the Application Server, "userid" is the corresponding Application Server login ID, and "password" is the user's password as shown in *Figure 8*.

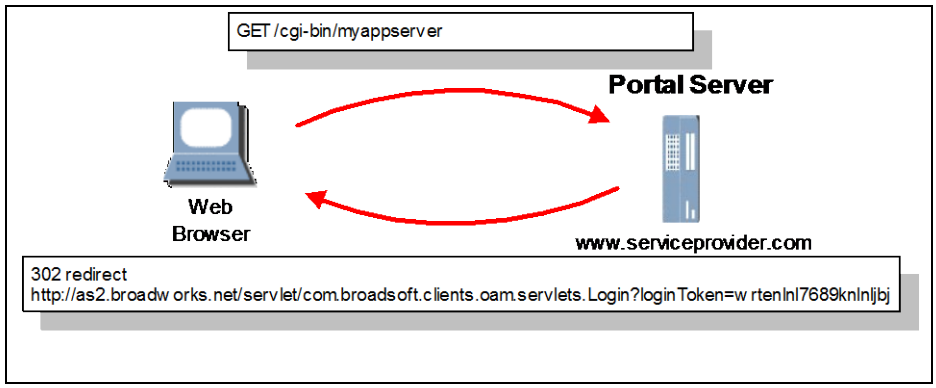

Figure 8 Interaction between End-user Browser and Portal Server with Automatic Login (User ID/Password)

NOTE: If the user has a blank password, which is possible with external authentication password rules, this approach does not work. A login token must first be prepared, as shown in steps 4 and 5.

In a replicated Application Server scenario, where Application Servers are in clusters, an additional parameter can shorten the login time, by instructing the Application Server not to verify when the user login should proceed on the specified machine. Otherwise, the Application Server applies the same login logic as described earlier in steps 2 through 6. This parameter is used only in conjunction with the *UserID/Password* parameters. In this case, the URL is:

```
http://ashost/servlet/Login
?UserID=userid&Password=password&noRedirect=true
```

[1] It is recommended when using the Auto-login feature of the API with the exposed user ID/password that the interactions between the browser and the portal server are done over a secure SSL channel.

4.7 Return User to Portal Server

Optionally, a *redirectURL* parameter can be added to the redirection URL. The Application Server web application uses the redirectURL to return the user's browser to the portal server if the Application Server web session expires. This is done to avoid showing the Application Server *Login* page, which would usually be shown under these circumstances. The complete URL would be similar to the following (assuming http://www.serviceprovider.com/portallogin?userId=dude is the redirectURL):

```
http://ashost/servlet/Login
?loginToken=preparedToken&redirectURL=
http%3A%2F%2Fwww.serviceprovider.com%2Fportallogin%3FuserId%3Ddude
```

or

```
http://ashost/servlet/Login
?UserID=userid&Password=password&noRedirect=true&redirectURL=
http%3A%2F%2Fwww.serviceprovider.com%2Fportallogin%3FuserId%3Ddude
```

4.8 Message Flow

The message flow in *Figure 9* shows the typical sequence of events when using the portal API on the Application Server for an end-user login.

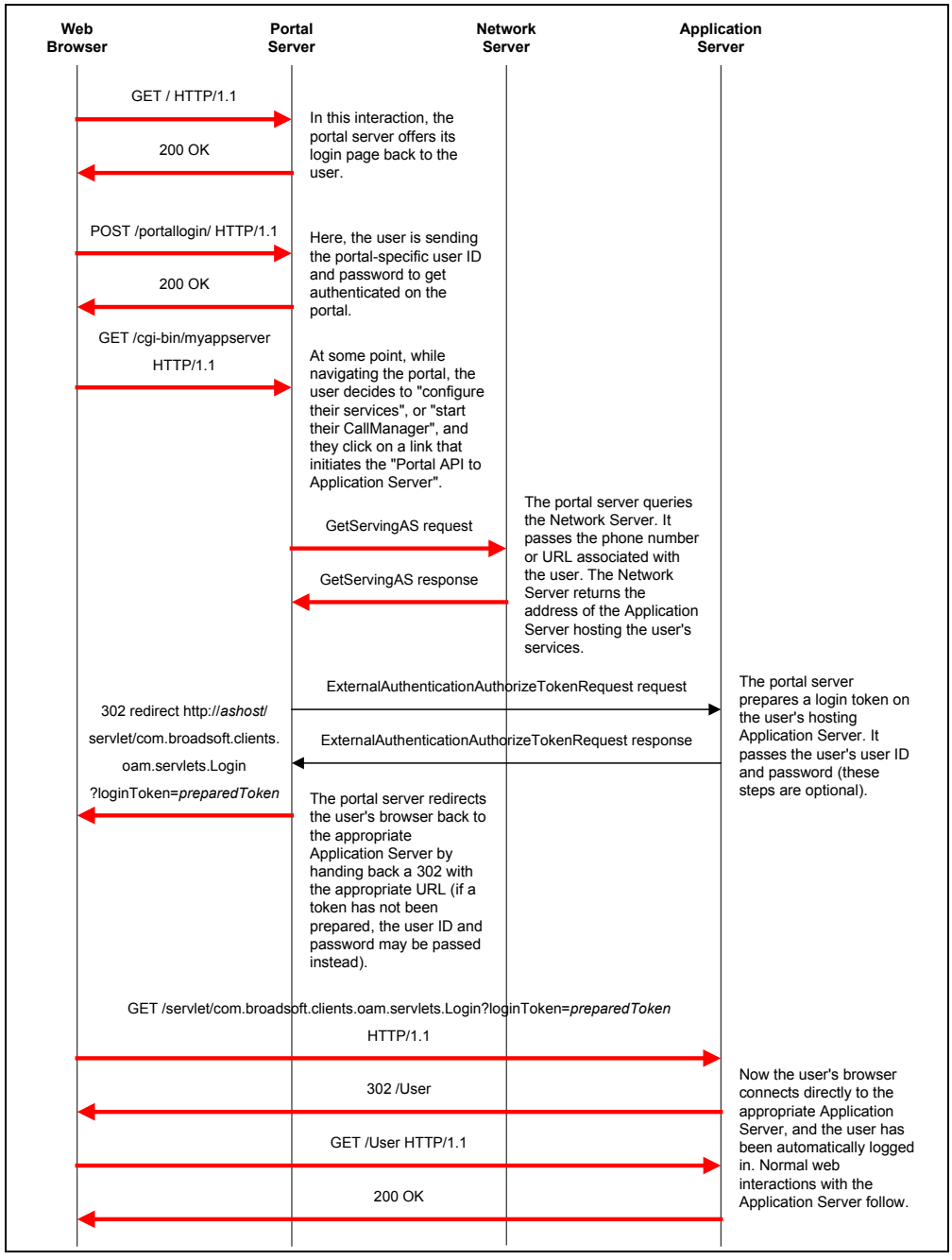

Figure 9 Portal API Message Flow for an End-user Login

5 Administrator Login on Non-replicated Application Server

The steps involved in using the location and login APIs for an administrator login on a non-replicated Application Server are as follows:

1) The administrator logs in to the portal server and navigates the web pages. The administrator clicks on a link to go to the Application Server configuration pages.
2) The portal server is aware of which Application Server the administrator is supposed to log into, or offers a list of choices to the administrator. The target Application Server is therefore identified implicitly or explicitly.
3) The portal server uses the location API on the identified Application Server to prepare a login token. This step is necessary to hide the user ID and password from the administrator (on the Application Server). If this is not a concern, this step can be skipped.
4) The Application Server returns a successful response.
5) The portal server redirects the administrator's browser to the Application Server with appropriate credentials (prepared token or user ID/password) to automatically log the administrator in. As part of this step, the portal server can optionally specify a redirect URL for the administrator when the Application Server web session expires. Usually, the administrator is presented with the Application Server *Login* page when the browser session expires.
6) The administrator's web browser accesses the login servlet on the Application Server with the provided parameters.

This sequence of interactions is shown in *Figure 10*.

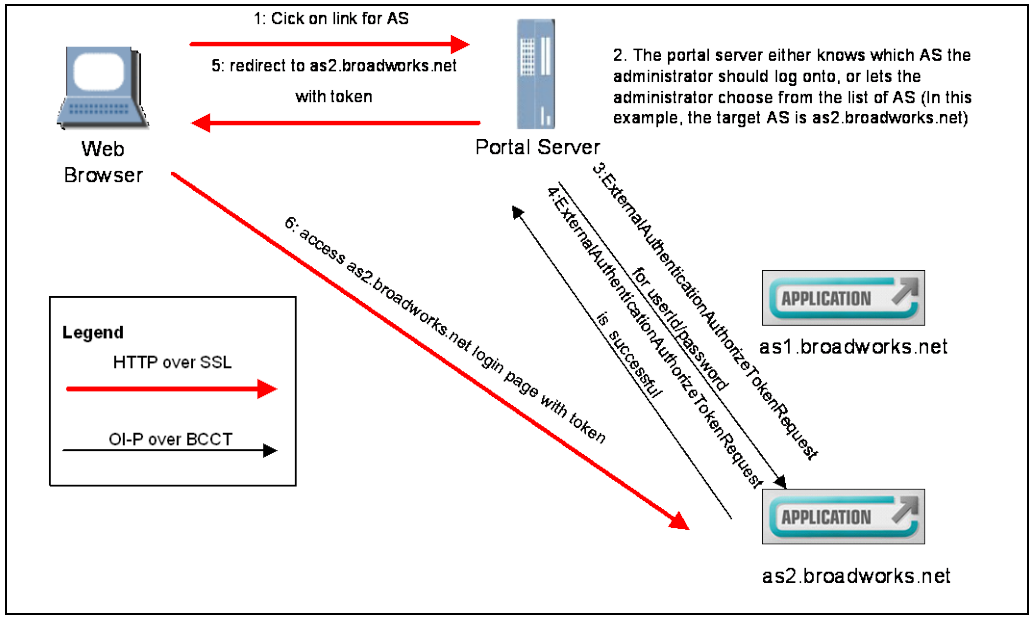

Figure 10 Administrator Login (Non-replicated Scenario)

The administrator has the option of being directed to the Application Server and presented with the standard Application Server *Login* page, or choosing to log in to the Application Server automatically.

5.1 Portal Side Administrator Profile

To use the API, the portal server must maintain a profile of attributes associated with each administrator who can access the Application Server configuration pages. These attributes, shown in *Table 2*, are used as parameters to each API.

Attribute Name	Attribute Type	API	Description
target Application Server	Address or array of addresses	Login	The portal server must either be aware of which Application Server the administrator should log in to, or offer the administrator a choice of Application Servers.
administrator ID	String	Login/Location	Application Server login ID associated with the administrator.
administrator password	String	Login/Location	Application Server password associated with the administrator's login ID. If external authentication password rules are enabled on the Application Server, and if the administrator has a blank password, then the password attribute value for the location API is an empty string. Blank passwords are not supported directly in the login API. A login token must be prepared using the location API, after which the administrator can log in through the login API using that token.

Table 2 Administrator Profile Attributes (Non-replicated Scenario)

5.2 Prepare Login Token

If the administrator's user ID and password on the Application Server must be hidden from the administrator, the location API must be used to prepare a login token before redirecting the administrator to the target Application Server.

For more details, see section *4.3 Prepare Login Token*.

5.3 Redirect Administrator to Login Page

After the portal server has identified the target Application Server either implicitly or explicitly, it can then redirect the administrator to that appropriate Application Server with the login API. For more details, see section *4.4 Redirect User to Login Page*, *4.5 Automatic Login with Token*, and *4.6 Automatic Login with User ID and Password*.

5.4 Return Administrator to Portal Server

Optionally, a *redirectURL* parameter can be added to the redirection URL. The Application Server web application uses the redirectURL to return the administrator's browser to the portal server when the web session expires. For more details, see section *4.7 Return User to Portal Server*.

5.5 Message Flow

The message flow in *Figure 11* shows the typical sequence of events when using the portal API for an administrator login on a non-replicated Application Server.

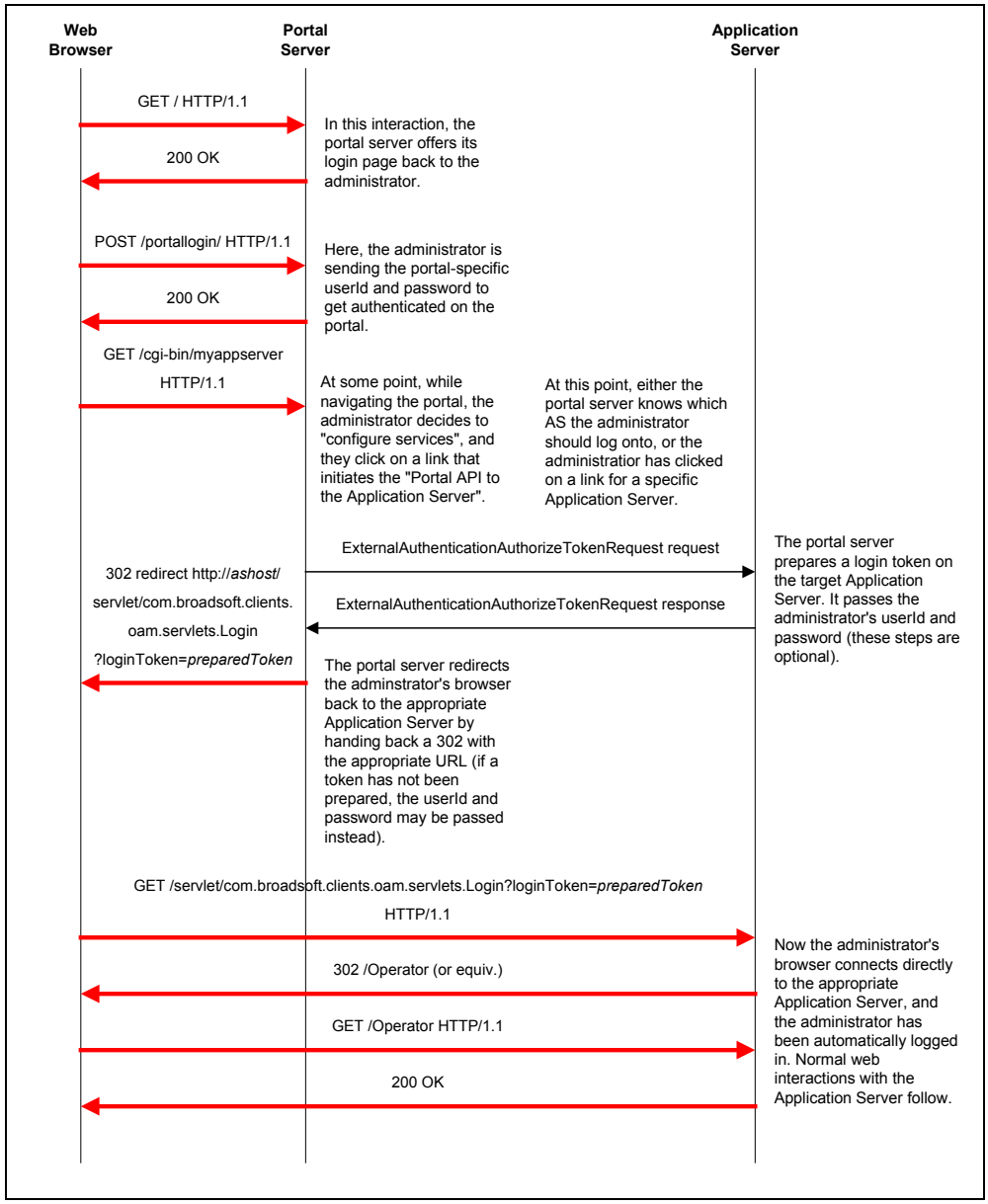

Figure 11 Portal API Message Flow for Administrator Login (Non-replicated Scenario)

6 Administrator Login on Replicated Application Server

The steps involved in using the location and login APIs for an administrator login on a replicated Application Server are as follows:

1) The administrator logs into the portal server and navigates the web pages. The administrator clicks on a link to go to the Application Server configuration pages. The portal server is aware of which Application Server cluster the administrator is supposed to log in to, or offers a list of choices to the administrator. The portal server must know the address of at least one Application Server in each cluster.
2) The portal server uses the location API on any Application Server in the identified cluster to resolve the hostname of the primary Application Server in the cluster. (Optionally, the portal server could be aware at all times of which Application Server is the primary one, and this step could be skipped. However, this is not recommended, because it could be prone to errors.)
3) The Application Server returns the address of the primary Application Server in the cluster.
4) The portal server uses the location API on the primary Application Server to prepare a login token. This step is necessary to hide the user ID and password from the administrator (on the Application Server). If this is not a concern, this step can be skipped.
5) The Application Server returns a successful response.
6) The portal server redirects the administrator's browser to the primary Application Server with appropriate credentials (prepared token or user ID/password) to automatically log the administrator in. As part of this step, the portal server can optionally specify a redirect URL for the administrator when the web session expires. Normally, the administrator is presented with the Application Server *Login* page when the browser session expires.
7) The administrator's web browser accesses the login servlet on the primary Application Server with the provided parameters.

This sequence of interactions is shown in *Figure 12*.

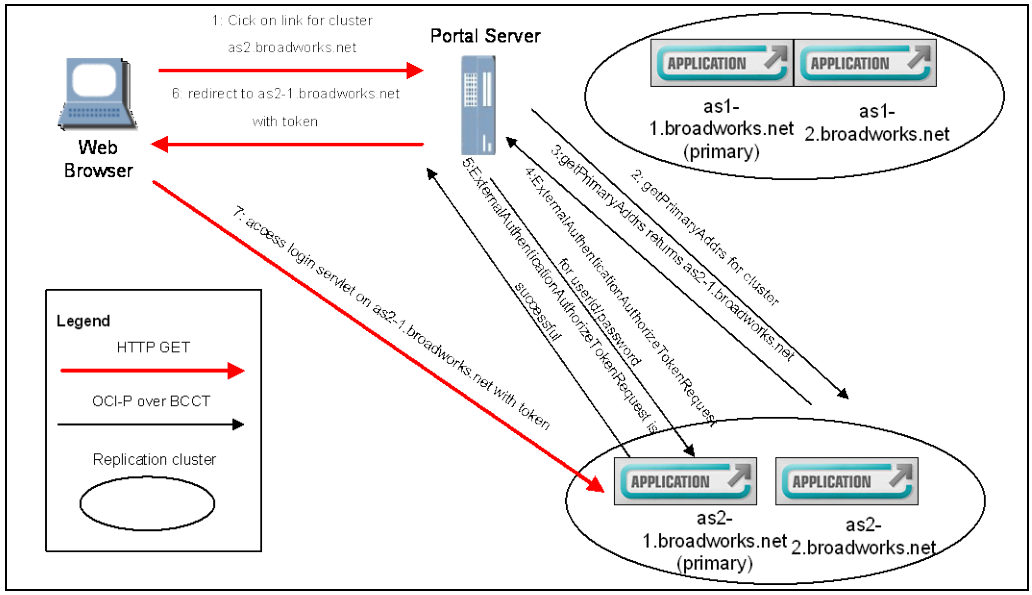

Figure 12 Administrator Login (Replication Scenario)

The administrator has the option of being directed to the Application Server and presented with the standard Application Server *Login* page, or choosing to log in to the Application Server automatically.

6.1 Portal Side Administrator Profile

To use the API, the portal server must maintain a profile of attributes associated with each administrator that can access the Application Server configuration pages. These attributes, shown in *Table 3*, are used as parameters to each of the APIs.

Attribute Name	Attribute Type	API	Description
Target Application Server cluster	Address or array of addresses	Location	The portal server must either be aware of which Application Server cluster the administrator should log on to, or offer the administrator a choice of Application Server clusters. For each Application Server cluster, the portal server must know the name of the cluster and the address of at least one Application Server in the cluster.
Administrator ID	String	Login/Location	Application Server login ID associated with the administrator.
Administrator password	String	Login/Location	Application Server password associated with the administrator's login ID. If external authentication password rules are enabled on the Application Server, and if the administrator has a blank password, then the password attribute value for the location API is an empty string. Blank passwords are not supported directly in the login API. A login token must be prepared using the location API, after which the administrator can log in through the login API using that token.

Table 3 Administrator Profile Attributes (Replication Scenario)

6.2 Obtain Address of Primary Application Server

The addresses of the primary server in the cluster can be obtained using the PrimaryInfoGetRequest OCI command of the Application Server location API. A sample interaction is shown in *Figure 13*.

For complete details on the location API, see the *Application Server Portal API Specification* [2], and for sample client implementations, see the *BroadWorks Network Server Provisioning Interface Specification* [3].

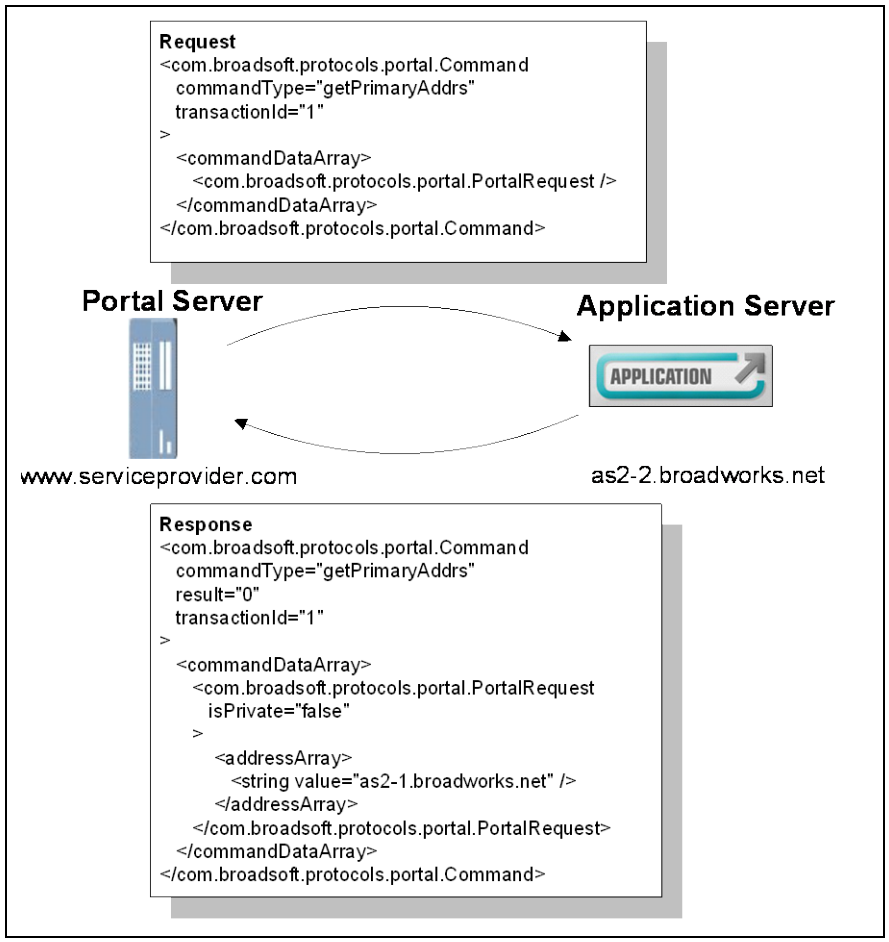

Figure 13 Location API on Application Server (PrimaryInfoGetRequest OCI Command)

6.2.1 Build Request Document

Before the portal server can use the location API, it must build a valid XML request document with an appropriate set of commands. The portal server must build and send a PrimaryInfoGetRequest OCI request, similar to the following example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<BroadsoftDocument protocol="OCI" xmlns="C"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <sessionId xmlns="">C888E4E012130AD3C8613A92FD24EFAE</sessionId>
  <command xsi:type="PrimaryInfoGetRequest" xmlns="">
    <isPrivate>false</isPrivate>
    <isAddressInfoRequested>true</isAddressInfoRequested>
  </command>
</BroadsoftDocument>
```

Inside the request, the portal server may include the optional element “isPrivate”. The example in *Figure 13* uses the default value of the *isPrivate* parameter (“false”):

If the Application Server is set up in a dual-homed configuration, with the web application available from both the public (access) and private (signaling) networks, the “isPrivate” attribute can be used to indicate the type of address to be returned. By default, public access addresses are returned.

If the Application Server has an external Web Server (see Release 10 External Web Server Support functionality), then the public address is the external Web Server address, while the private address is the address of the collocated Web Server on the Application Server host.

6.2.2 Process Request

The Application Server processes the request and returns the primary server address(es).

6.2.3 Decode Successful Response Document

If the Application Server processes the request successfully, it returns an OCI *PrimaryInfoGetResponse*. Any other response indicates an error. A successful response is similar to the following:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<BroadsoftDocument protocol="OCI" xmlns="C"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <sessionId xmlns="">C888E4E012130AD3C8613A92FD24EFAE</sessionId>
  <command debugInfo="5:15" echo="" xsi:type="PrimaryInfoGetResponse"
xmlns="">
    <isPrimary>true</isPrimary>
    <hostnameForPrimary>AS2</hostnameForPrimary>
    <addressForPrimary>as2-1.broadworks.net</addressForPrimary>
  </command>
</BroadsoftDocument>
```

In the above example, the response has one “addressForPrimary” element set to the value of “as2-1.broadworks.net”. This is the (public) address of the primary Application Server in the cluster. For certain configurations (for example, multi-path), there can be more than one “addressForPrimary” and “privateAddressForPrimary” elements returned. The portal server should always try to connect to the first address in the list, and try subsequent addresses only if a connection to the first address cannot be established.

6.2.4 Decode Failure Response Document

It is possible that the request may fail. This could happen for instance, if there is no communication with the Network Server. Although in this scenario the portal server does not communicate with the Network Server directly, a functioning Network Server must be reachable at all times from every Application Server in a replication scenario. The portal server should check the response, to determine if the request has succeeded.

When there is a failure, the response is of type “ErrorResponse” and elements “summary”, “summaryEnglish”, and “detail” explain the reason of the request failure. Receiving an error such as this usually indicates a provisioning problem either on the Application Server or on the portal server. How the portal server handles this error depends on the systems integrator.

6.3 Message Flow

Now that the address of the primary Application Server was obtained, the remainder of the interactions between the administrator's browser, the portal server, and the primary Application Server are the same as for the non-replicated scenario. For more information, see sections starting at *5.2 Prepare Login Token*.

The message flow in *Figure 14* depicts the typical sequence of events for the portal API, for an administrator login on a replicated Application Server.

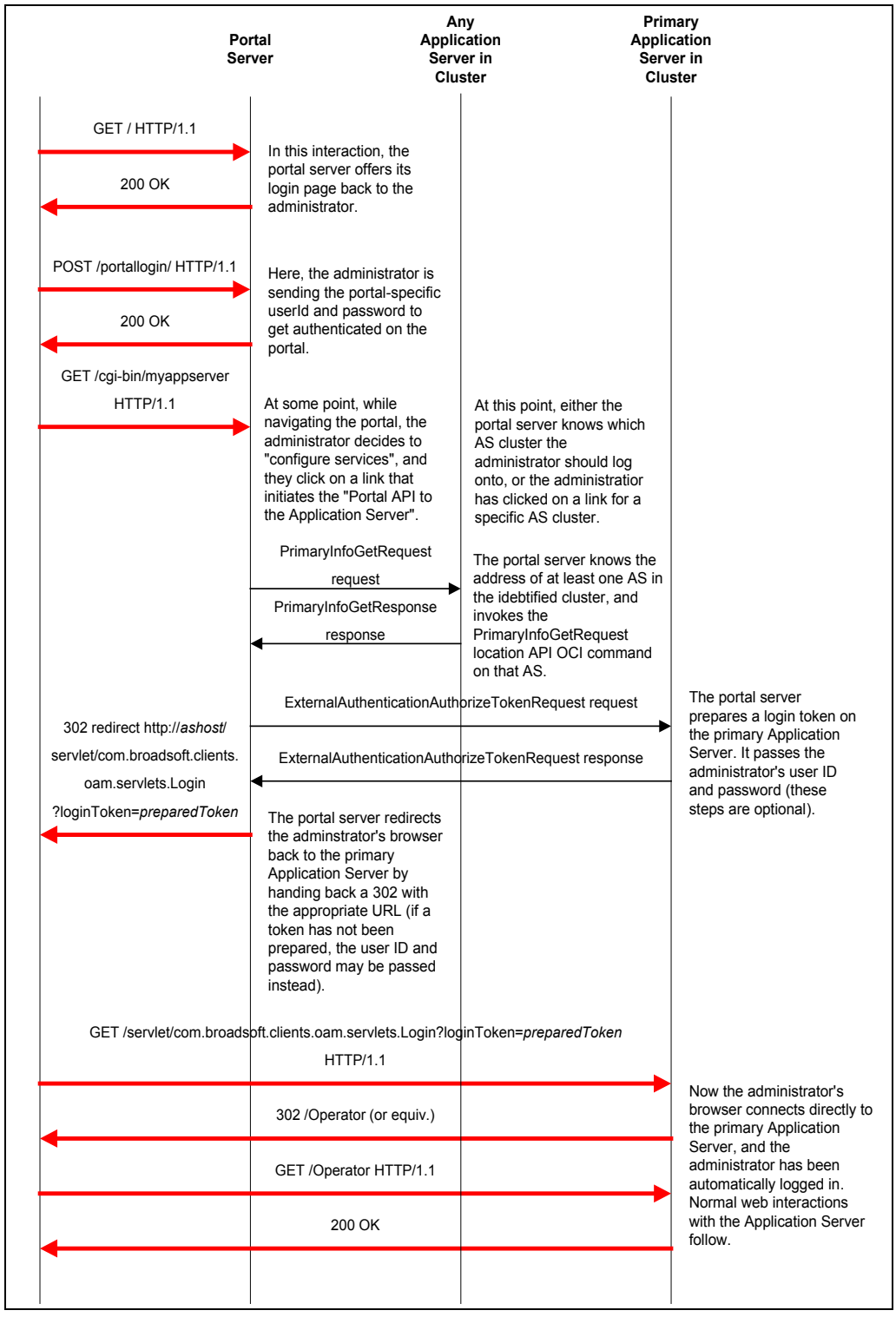

Figure 14 Portal API Message Flow for Administrator Login (Replicated Scenario)

7 Application Server Login API Specification

The login API on the Application Server is a URL (servlet) that accepts several *CGI* parameters when accessed via a HTTP Get or Post request. The servlet URL is of the following format:

```
http://ashost/servlet/Login
```

NOTE: If "full" SSL support has been enabled on the Application Server whether at installation or afterwards using the config-network utility, the above URL should specify the https:// protocol, especially if the portal web site is SSL-protected. Not specifying the appropriate protocol results in an unnecessary browser warning such as: "You are being redirected to a non-secure page".

If no parameters are present or if the auto-login fails, the Application Server redirects to the standard *Login* page (unless redirectURL is present, in which case that URL is used instead of the *Login* page). Optional parameters that can be passed using Get or Post requests are shown in *Table 4*.

Parameter	Type or Values	Description
loginToken	String (URL-encoded)	Allows for automatic login without exposing the user ID and password in the HTTP request. This string maps internally to a user ID/password pair, provided that the location API command ExternalAuthenticationAuthorizeTokenRequest has been successfully invoked.
UserID	String (URL-encoded)	Allows for automatic login in conjunction with the *Password* parameter. This is the user's or administrator's login ID.
Password	String (URL-encoded)	Allows for automatic login in conjunction with the *UserID* parameter. This is the end user's or administrator's password. If external authentication password rules are enabled on the Application Server, and if the end user or administrator has a blank password, then the password attribute value for the location API is an empty string. Blank passwords are not supported directly in the login API. A login token must be prepared using the location API, after which the end user or administrator can log in through the login API using that token.
noRedirect	true/(false or absent)	Used only in conjunction with the *UserID* and *Password* parameters to allow for a faster login, knowing that the portal server has already determined the correct hosting or primary Application Server for the end user or administrator. The default value is "false", except when the *loginToken* parameter is used, in which case this parameter is ignored if present, and the value is considered to be "true".

Parameter	Type or Values	Description
redirectURL	URL (URL-encoded)	Specifies a URL where the end user or administrator is redirected when the web session expires, rather than being redirected to the *Login* page.
isOriginatorPrivate	(true or absent)/false	This parameter, in conjunction with the login authorization level functionality, allows the portal server to lower the administrator's access level to “public”, even if the administrator is logging in from the “private” network. The default value is “true”, in which case the Web Server determines the real value of the access level (public/private) based on address comparison. However, if there is a need to have certain browser clients on the “private” network treated as if they were on the “public” network. The portal server could implement an access control list and tag all those clients as “public”, by setting the value of this parameter to “false”, which would force the Web Server to see those clients as if they were on the “public” side in spite of their real address. Providing an incorrect value for isOriginatorPrivate can only reduce the possible authorization level. Public addresses are also known as “access” addresses, while private addresses are also known as “signaling” addresses. “DualRouting” addresses are public and private at the same time.
rememberPass	on/(other string or absent)	Used in conjunction with the *UserID* and *Password* parameters, if the value is “on”, this parameter causes a cookie to be created, storing the user ID and password on the user’s machine. The user is able to subsequently log in by going to the `http://ashost/servlet/Login` URL.

Table 4 Login API CGI Optional Parameters

References

[1] BroadSoft, Inc. 2014. *Network Server Portal API Specification, Release 21.0.* Available from the BroadSoft Xchange at xchange.broadsoft.com.

[2] BroadSoft, Inc. 2014. *Application Server Portal API Specification, Release 21.0.* Available from the BroadSoft Xchange at xchange.broadsoft.com.

[3] BroadSoft, Inc. 2014. *BroadWorks Network Server Provisioning Interface Specification, Release 21.0.* Available from the BroadSoft Xchange at xchange.broadsoft.com.

[4] BroadSoft, Inc. 2014. *Application Server Provisioning Interface Specification Guide, Release 21.0.* Available from the BroadSoft Xchange at xchange.broadsoft.com.