Parallels Remote Application Server

Integrating with Parallels Clients

Parallels International GmbH
Vordergasse 59
8200 Schaffhausen
Switzerland
Tel: + 41 52 672 20 30
www.parallels.com

Contents

Introduction **4**

RAS Web Client API **5**

API Basics 5

Main URL Format 6

JSON Payload 6

Example 7

Additional Information 9

Direct App access 9

Parallels Client URL Scheme **11**

URL Scheme Basics 11

URL Format 12

Commands and Options 13

Get Parallels Client Version (GetVersion) 13

Launch Published Resources (LaunchApp) 14

Launch Resources with Advanced Security (Get2xa) 16

Log User Off (LogOff) 20

Index **21**

CHAPTER 1

Introduction

This guide is intended for Parallels® RAS customers and partners, such as Independent Software Vendors (ISVs), who intend to utilize a third-party web portal for authenticating their users and launching remote applications, desktops, and other published resources hosted in their Parallels RAS environment. Such an implementation is possible by integrating a custom solution to interact directly with Parallels Clients, including Parallels Client for Windows / macOS / Linux / iOS / Android, and the RAS Web Client.

Integrating a custom solution with Parallels Clients is done via the following interfaces available in Parallels RAS:

- **RAS Web Client API** — provides connection, user authentication, and resource launching methods called from a web browser via the RAS Web Client.
- **Parallels Client URL Scheme** — a custom URL scheme that allows you to perform actions in Parallels Client installed on a user device. Actions include configuring a connection, authenticating a user, and launching published resources.

The rest of this guide explains how to use the interfaces described above.

CHAPTER 2

RAS Web Client API

This chapter describes the RAS Web Client.

In This Chapter

API Basics 5
Main URL Format 6
JSON Payload 6
Example 7
Additional Information 9
Direct App access 9

API Basics

Note: To use the RAS Web Client, the **Enable User Portal** option must be enabled for the RAS Secure Gateway in the Parallels RAS Console and the gateway must be reachable on `https://<server>/userportal/` where `<server>` is the IP address or FQDN of the RAS Secure Gateway server.

Parallels Web Client is a browser-based app launcher that is available out of the box with Parallels RAS. The Parallels Web Client is described in detail in the Parallels RAS Administrator's Guide, which is available for download on the Parallels website at the following location: https://www.parallels.com/products/ras/resources/.

The interface described here makes it possible to implement a custom resource launcher.

The following is a typical scenario of the RAS Web Client usage:

1. A user logs in to the local web portal where the available published resources are listed.
2. The user clicks on a resource (e.g. an application) that is to be served via Web Client.
3. A new tab is opened in the web browser in the background using the RAS Web Client URL and some additional parameters, one of which is the name of a JSON payload containing user credentials and the information about the resource being launched. The customer or partner is expected to generate the required payload (we'll talk more about it later in this section).
4. The Secure Gateway loads the Parallels Web Client and performs a GET request to obtain the JSON payload (see above).

5 The RAS Web Client then calls a function passing the credentials and resource information.

6 The user is authenticated and, if successful, the resource is launched on the user's desktop.

Main URL Format

When a user clicks a resource name (a link) on a custom web page, the URL behind it must have one of following formats:

```
https://<server>/userportal/?appinfo=https://<webserver>/<app-name>.js&
theme=<theme-name>#/launch
```

or

```
https://<server>/<theme-name>/?appinfo=https://<server>/<app-name>.js#/
launch
```

where:

- `<server>` is the IP address or FQDN of the RAS Secure Gateway server.
- `<webserver>` is a web server from where information about published resources and authorized users is served.
- `<app-name>.js` is a payload in JSONP format containing user credentials and a published resource information. Note that since payload must contain user credentials, the payload should be generated dynamically for every user and every application, desktop, published resource to which a user has access through the web portal. For testing purposes, you can create a temporary ".js" file with all the required information and save it in a folder on your web server. The payload structure is described in the section that follows this one.
- `<theme-name>` is the name of the Theme that will be used. This parameter is optional if you use the first URL format.

Note: You can use the `/RASHTML5Gateway` suffix instead of `/userportal`, but this is not recommended.

JSON Payload

The JSON payload that the RAS Secure Gateway receives as a response has the following structure (the values are for demonstration purposes only and should be substituted with your own):

```
_RASWebClientLoadApp({
   u: 'username',
   q: 'password',
   a: '#4',
   p: 'c:\\temp\\another.txt',
   extra: {
```

```
    redirectPrinter: true,
    redirectLinks: true,
    redirectSound: true
  }
});
```

Note: You can also use the older function `_RASHTML5LoadApp` instead of `_RASWebClientLoadApp` for backwards compatibility.

The following table describes the available JSON parameters:

Parameter Name	Description
u:	User name
q:	User password
a:	Published resource ID, as displayed in the RAS Console in the **Publishing** category. It's the number in front of the application name on the **Information** tab (e.g. #4: Google Chrome). The number must be included in the payload together with the pound sign, e.g. "#4". If this parameter is omitted, the user will see all applications available on User Portal.
p:	Optional arguments (if an application has them).
d:	Domain name.
extra: redirectLinks:	Boolean: true = redirect links; false = do not redirect.
extra: redirectPrinter:	Boolean: true = redirect printer; false = do not redirect.
extra: redirectSound:	Boolean: true = redirect sound; false = do not redirect.

Example

In the following example we'll put together a very simple browser-based application launcher that uses the RAS Web Client.

Before we begin, let's make sure that our RAS Secure Gateway configuration meets the requirements:

1. In the RAS Console, navigate to **Farm** > <Site> > **Gateways**.
2. Right-click a Gateway and click **Properties**.

3 In the **Properties** tab, make sure that the **Enable RAS Secure Gateway in site** option is selected.

4 In the **User Portal** tab, make sure that the **Enable User Portal** option is selected.

5 Make sure that the Web Client opens in your default web browser.

We are now ready to create our custom resource launcher. In this example, we are making the following assumptions:

- Our Secure Gateway is running on the server with the following IP address: 192.168.10.10.
- Our web server is running on the server 192.168.20.20.
- The published resource that we'll be launching is Google Chrome and it's ID the in the RAS Console is "#4".

  You can look up the ID in the RAS Console (**Publishing >** select a resource **> Information tab >** look at the first field on the tab page, which displays the resource ID followed by resource name). Note that the pound sign (#) must be included together with the actual ID. You can also obtain a resource ID via RAS PowerShell by executing `Get-RASPubItem "resource-name"`. The returned `PubItem` object has the Id property that specifies the resource ID. To get the list of all available published resources, execute the `Get-RASPubItem` cmdlet with no parameters.

First we need to create a JSON payload containing user credentials and the application info. To do so, open a text editor and type (or paste) the following:

```
_RASWebClientLoadApp ({
    u: 'user-name',
    q: 'user-password',
    a: '#4',
    p: '',
        extra: {
            redirectPrinter: true,
            redirectLinks: true,
            redirectSound: true
        }
});
```

In the payload above, substitute 'user-name' and 'user-password' with your own. Change the "a:" parameter value to the ID of the published application that you wish to launch. Save the payload as a text file with the ".js" extension in the folder on your web server. In our example, we'll save the file as Google-Chrome.js in the wwwroot folder of our web server.

We now need to create a web page from where users will be launching our published application:

1 Create a simple HTML page and save it to a folder on a local web server from where it can be opened in a web browser.

2 Add the following link to the web page and name it "Google Chrome" (or use the name of your own application):

```
https://192.168.10.10/userportal/?https://192.168.20.20/Google-Chrome.js#/launch
```

In the URL above, substitute the Gateway address (192.168.10.10) and the web server address (192.168.20.20) with your own, and use the ".js" file name that you created in previous steps.

3 Save the web page.

To test our custom launcher:

1 Open the web page in a browser.

2 Click the Google Chrome link (or the name you used).

3 The remote application will open in the browser using Parallels Web Client.

Additional Information

This section describes how the API works behind the scenes. You cannot change any of it because that's how Secure Gateways operate, but it should give you an idea of what is involved.

1 When you open the main URL in a browser (`https://<server>/userportal/?https://<webserver>/<app-name>.js#/launch`), you invoke the Secure Gateway.

2 The Gateway loads the Parallels Web Client and, using the second part of the URL above, performs a GET request with a `<script>` tag (this overcomes issues when working across varying domains):

```
<script type='text/javascript'
src='https://webserver.host/getAppInfo/JSONP/7EB8A5D1C2D3E986AB432D'
></script>
```

3 RAS Web Client obtains the JSON payload and then calls the following Gateway function passing the parameters from the JSON payload:

```
_RASWebClientLoadApp ({ u: 'user-name', q: 'user-password', a: '#4', p:
'c:\\temp\\another.txt', extra: { redirectPrinter: true, redirectLinks:
true, redirectSound: true } });
```

4 The function call above launches the resource in the browser using Parallels Web Client.

Direct App access

Starting with Parallels RAS 18, specific published resources may be directly accessed through RAS Web Client. This can be achieved with the introduction of a new parameter, *appid*, which allows administrator to use links to access the published resource directly. This allows a more flexible and easier way for users to access Parallels RAS published resources such as using browser shortcuts or bookmarks or third-party portals such as Azure My Apps Portal to access independent SAAS applications and Parallels RAS virtual apps and desktops.

To launch a published resource directly, you need to specify a URL using one of the following formats:

URL format	Description
https://<server>?appid=	This format omits the Theme name and uses the default Web Client Theme. The "appid" parameter specifies the published resource ID as seen in the Publishing category in the RAS Console. The ID is automatically generated when a resource is published. To see it, select a published resource and examine the **Application** field on the **Information** tab. For example, #5: Microsoft Office Word — the application ID of the Microsoft World here is 5.
https://<server>/<theme-name>?appid=<app-ID>	This format is similar to the one above, but specifies a Theme name.
https://<server>/userportal?theme=&appid=	This format is the same as the one above, but uses the full URL specification. It is listed here just for reference.

Supported parameters:

Parameter	Description
appid	The published item (application or desktop) to be launched. Note: the format is just a number without # in front.
overrideparams	[Optional]. URL Encoded Override arguments that needs to be passed to the published application.

Example:

https://<server>?appid=14&overrideparams=C%3A%2Ftest.txt

When opening a published resource using a direct link, the **Auto login** option is also used depending on the settings.

CHAPTER 3

Parallels Client URL Scheme

This chapter described the Parallels Client URL scheme.

In This Chapter

URL Scheme Basics..... 11
URL Format 12
Commands and Options 13

URL Scheme Basics

Parallels Client URL scheme allows you to perform actions and programmatically interact with the Parallels Client installed on a user device. The actions include the following:

- Automatic configuration of Parallels RAS or RDP connection in Parallels Client using predefined settings.
- User Authentication.
- Launch of published resources (application, desktop, document, etc.) from a web page or another application.
- User session log off from a specific RAS server of from all current sessions on all servers.

There are currently two URL schemes used in Parallels RAS: `tuxclient://` and `prlclient://`. Depending on the platform on which Parallels Client is running, the following URL schemes are used:

Platform	URL scheme
Windows	`tuxclient://`
Linux	`tuxclient://`
macOS	`tuxclient://`
iOS	`tuxclient://` and `prlclient://`
Android	`tuxclient://` and `prlclient://`
Chrome	Not supported

Note that the `tuxclient` scheme is kept for backwards compatibility with older Parallels Clients. In the future, the `prlclient` scheme may replace it in all versions of Parallels Client. At this time, use the scheme that is supported on a given platform. If both are supported, you can use one or the other -- the usage, including command options, is exactly the same.

When Parallels Client is installed on a device, it registers the Parallels Client URL scheme with the operating system, which tells a web browser what to do when the user opens such a URL. In this instance, the web browser will open Parallels Client passing to it the parameters that the URL contains. Parallels Client will process the parameters and will perform actions according to received instructions.

The following describes a typical Parallels Client URL scheme usage scenario:

1. Suppose you want to create an application hub or a web portal from where users can launch applications published in Parallels RAS.
2. You first compose a URL according to specifications, which are described in detail later in this chapter. The URL string begins with the Parallels Client URL scheme followed by the necessary parameters depending on the task and configuration options. For example, `prlclient:///?Command=LaunchApp&AppID=2360&...` (the complete URL is not shown for brevity).
3. You then publish the URL on a third-party web portal where end users can use it to launch a published resource.

A user logs in to the web portal and clicks the URL. This opens Parallels Client, which processes the parameters contained in the URL and performs actions according to these parameters (configures a connection, authenticates the user, launches an app, etc.).

Note that if Parallels Client is not installed on a user device, nothing will happen because the URL scheme is unknown to the operating system and no application is associated with it. This only means that Parallels Client must be installed before making use of the custom scheme URL.

URL Format

A URL string always begins with the Parallels Client URL scheme. See the table in the previous section for a scheme supported on a specific platform. If both URL schemes are supported, you can use either one.

The first parameter that you include after the URL scheme is the `Command` parameter. It tells Parallels Client what kind of action should be performed. The available commands are:

- **GetVersion** (p. 13) — Allows to determine on the web server side which Parallels Client version is installed on a user device, or if the Parallels Client is installed at all.
- **LaunchApp** (p. 14) — Creates a connection, authenticates a user, launches a published resource.

- **Get2xa** (p. 16) — Performs the same actions as the LaunchApp command (above), but uses advanced security, which eliminates passing sensitive information (server name, user credentials, published resource info) in the URL itself.
- **LogOff** (p. 20) — Logs off the user from a session on a specified server or from all sessions on all servers.

The `Command` parameter is placed in the URL as shown in the following example:

```
tuxclient:///?Command=LaunchApp
```

Command options are specified after the command using the ampersand ("&") separator:

```
tuxclient:///?Command=LaunchApp&AppID=2360&ConnMode=0 ...
```

Note that the `prlclient://` URL scheme uses the same exact format demonstrated here and in all other examples included in this chapter.

Commands and Options

Custom types

The following custom types are used in command options described in this chapter:

Type	Possible values	Description
Boolean	YES, NO	The values are case sensitive.

Get Parallels Client Version (GetVersion)

The **GetVersion** command allows to determine on the web server side which Parallels Client version is installed on a user device, or if the Parallels Client is installed at all. If Parallels Client is an older version or not installed on a device, you can display a message to the user with a link from which they can download and install the correct version of Parallels Client.

The command works as follows:

1. A user opens a custom scheme URL with the **GetVersion** command in it.
2. The URL starts Parallels Client which receives the parameters contained in the URL, including server name, port number, the web portal session ID, and the path to which Parallels Client needs to connect to pass its version information back to the web portal.
3. The Parallels Client uses the received information and connects to the web portal using the specified path with the version information appended to it.

4 The web portal parses the received data and evaluates the Parallels Client version number. If the request times out (no response is received from Parallels Client), it means that Parallels Client is not installed on the user device.

The following table describes the **GetVersion** command options:

Key	Datatype	Value
Command	String	The name of the command to execute. In this instance, it's "GetVersion".
Server	String	The server on which the web portal is hosted. Server can be specified by IP address or its local name.
Port	Int	The port number of the web portal
Session	String	The web portal Session ID.
Secure	Boolean	The SSL boolean value.
Path	String	The path to which Parallels Client needs to connect in order to pass the version information. The OS type and the Parallels Client version info need to be specified with the path: **OS:** • **Key:** os • **Type:** Int • **Value:** 0 (Windows); 1 (Linux); 2 (macOS); 4 (Android); 5 (iOS). **Version:** • **Key:** version • **Type:** String • **Value:** The Parallels Client version. This needs to be URL encoded.
RequestPage	String	The exact path of the current web portal page from where the URL was opened. This is used in iOS, so that the iOS client can redirect the user back to the RAS Web Portal page.

URL example

```
tuxclient:///?Command=GetVersion&Server=my.server.testing&Port=80&Session=adm2dpjq3jomvk45mzktuy45&Secure=NO&Path=%2fmywebportal%2fTuxClientDetection.aspx%3fapp%3d2XClient%26installed%3d1&RequestPage=http://my.server.testing/mywebportal/Logon.aspx
```

Launch Published Resources (LaunchApp)

The **LaunchApp** command is used to configure a connection in Parallels Client, authenticate a user, and launch published resources.

Note that the **LaunchApp** command passes sensitive information inside a URL, including server name, port, and possibly username and password. If this is a concern, use a more secure **Get2xa** command (p. 16).

The following table describes the **LaunchApp** command options:

Key	Datatype	Value
Command	String	The name of the command to execute. In this instance, it's "LaunchApp".
AppID	Int	The application ID to launch. You can look up the ID in the RAS Console (**Publishing** > select a resource > **Information** tab > look at the first field on the tab page, which displays the resource ID followed by resource name). Note that the ID must be specified *without* the pound sign (#). You can also obtain a resource ID via RAS PowerShell by executing `Get-RASPubItem "resource-name"`. The returned `PubItem` object has the `Id` property that specifies the resource ID. To get the list of all available published resources, execute the `Get-RASPubItem` cmdlet with no parameters. If this parameter is empty, Parallels Client will perform an application listing.
MultiID		A comma separated list of applications to be launched by the client. This is used for the 'Start On Logon' option. If this is present, AppID will be ignored. **Note:** Currently not used by mobile clients.
Alias	String	Connection name (when creating a new connection).
ConnType	Int	Connection type: • 0 — Parallels RAS Connection • 2 — Standard RDP Connection
ConnMode	Int	Connection mode: • 0 — Gateway • 1 — Direct • 2 — Gateway SSL • 3 — Direct SSL
Server	String	The server FQDN or IP address.
Backup	String	The secondary connection server (if available and required).
Port	Int	The port number.
UserName	String	Deprecated, but still supported for backwards compatibility. For new integrations, use LoginEx. User name. If this parameter is empty, the user will be prompted to enter a name.
LoginEx	String	User name. If this parameter is empty, the user will be prompted to enter a name. Usernames can have 'username', 'domain\user', or 'username@domain' format.
Password		Plain text password. If neither this key nor the “EncPass” (see below) is included, the user will be prompted for password when connecting.

EncPass	String	Encrypted ('hashed') password. If neither this key nor the “Password” is included, the user will be prompted for password when connecting.
SessionID	String	The Auth Session ID.
Connect	Boolean	Whether to connect to Parallels RAS right after the connection is configured. "YES" -- connect. "NO" -- don't connect (this should be used when you only want to create a connection without launching a published resource). Note that the values are case sensitive.
Save	Boolean	Whether to save the connection in Parallels Client. If set to "YES", the connection information will be saved (possibly overwriting existing settings). The default value is "NO". Note that the values are case sensitive.
Request Page	String	[Optional] The exact path of the web portal page from where the original URL was launched. This parameter is used on iOS devices, so that the iOS client can redirect the user back to the original web portal page after closing the application.
SSO	String	[Optional] Custom 3rd party SSO GUID. Used only by Windows clients. Currently only used by invitation emails.
HelpDeskEmail	String	[Optional] Helpdesk or support contact email. Currently only used by Parallels Clients for Android and iOS.
OverrideArgs	String	Override application arguments. Currently only used by Parallels Client for Mac.

URL example

The following URL creates a new connection (but doesn't save it in Parallels Client as the "Save" key is set to "NO"), authenticates a user using the specified credentials, and launches an application in Parallels Client.

```
tuxclient:///?Command=LaunchApp&AppID=2360&Alias=My-RAS-Connection&ConnType=0&ConnMode=
0&Server=10.0.0.50&Backup=&Port=80&LoginEx=tester56%402x&EncPass=l6Y1%2bfOryvNHYsI0nWIW
4g%3d%3d&SessionID=A78399FE-7077-43AD-8D0A-03641F1C759B&Connect=YES&Save=NO&RequestPage
=http://my.portal.testing/mywebportal/Dashboard.aspx
```

Launch Resources with Advanced Security (Get2xa)

The **Get2xa** command performs the same tasks as the **LaunchApp** command (p. 14), but uses an advanced security mechanism to pass sensitive information between the web portal and the Parallels Client.

Advanced security is achieved as follows:

1. A user clicks a published resource on a third-party web portal. This opens a URL that uses the Parallels Client URL scheme and includes the following information:
   - The web portal server name, port number, and session ID.

- A path from which Parallels Client can download an XML file containing the Parallels RAS connection information, user credentials, and the ID of the published resource to launch.

  The XML file that the Parallels Client will be downloading is called a 2XA file, which is a historical name used in Parallels RAS to identify the specific file format. The 2XA file specifications (XML) are described later in this section.

2. The URL opens Parallels Client in the background. The information contained in the URL is passed to Parallels Client.
3. Parallels Client connects to the web server using the received information and downloads a 2XA file using the path that it received via the URL.

   Note that the 2XA file should be dynamically generated for every user and every published resource when a given user attempts to launch a resource in the web portal.
4. Parallels Client parses the information contained in the XML file and uses it to create a connection, authenticate a user, and launch a resource.

Get2xa command options

The following table describes the **Get2xa command** options:

Key	Datatype	Value
Command	String	The name of the command to execute. In this instance, it's "Get2xa".
Server	String	The server where the web portal is hosted. The server can be specified by IP address or its local name.
Port	Int	The port number.
Session	String	The web portal session ID. This is used by Parallels Client in the GET request. It is being passed as a cookie named "ASP.NET_SessionId" (see **The GET request from Parallels Client** below).
Secure	Boolean	The SSL boolean value. "YES" - use SSL. "NO" - don't use it. The values are case sensitive.
Path	String	A path to which Parallels Client needs to connect in order to download the 2XA file. This needs to be URL encoded.
RequestPage	String	[Optional] The exact path of the current web portal page from where the original URL was launched. This is used in iOS, so that the iOS client can redirect the user back to the web portal page.

URL example

The following URL opens Parallels Client and passes to it the web server information, the web portal session ID, the path to the 2XA file, and the path to the original web portal web page.

```
prlclient:///?Command=Get2xa&Server=my.server.testing&Port=80&Session=adm2dpjq3jomvk45mzktuy45&Secure=YES&Path=%2fmywebportal%2fDashboardSource.aspx%3frApplicationID%3d2360%26rtype%3dget%26rcommand%3drun_secure_app%26rfarm%3d3&RequestPage=https://my.server.testing:80/mywebportal/Dashboard.aspx
```

The GET request from Parallels Client

When Parallels Client receives the information from the URL, it uses it to connect to the web server and download the 2XA file.

The following is an example of the GET request performed by Parallels Client:

```
GET %%Path%% HTTP/1.1
Host: %%Server%%:%%Port%%
User-Agent: RAS Client
Content-Type: application/x-www-form-urlencoded
Cookie: ASP.NET_SessionId=%%Session%%
```

2XA XML details

The following table describes the XML document structure used in the 2XA file (see also the XML example below):

Section	Key	Description
IIS	ValidSession	This should be set to 1.
Logon	User	Deprecated, but still supported for backwards compatibility. For new integrations, use LoginEx. The username.
Logon	LoginEx	The username. Usernames can have 'username', 'domain\user', or 'username@domain' format.
Logon	Base64ClearPassword	[Optional] Base 64 encoded plain text password. If this is not set, the Parallels Client will try to load the ClearPassword parameter (see below), else the user will be prompted for password when connecting. The parameter is available in all Parallels Client versions (desktop and mobile) since RAS v16.5.2.
Logon	ClearPassword	[Optional] Plain text password. If this is not set, the user will be prompted for password when connecting. This parameter has been available in Parallels Client for Windows for some time. It is available in Parallels Client for other platforms (desktop and mobile) since RAS v16.5.2.
Logon	SSO	[Optional] The Authentication mode. Can be one of the following: 0 = credentials (default) 3 = web
Connection	Port	The server's port number.
Connection	StartMode	The connection mode. Can be one of the following: 0 : Gateway Mode 1 : Direct Mode 2 : Gateway SSL 3 : Direct SSL

Connection	PrimaryServer	The server name. Server can be specified by IP or by its local name.
Startup	PublishedApp	[Optional] The ID of the published resource to be launched, including the pound sign (e.g. "#256"). You can look up the ID in the RAS Console (**Publishing** > select a resource > **Information** tab > look at the first field on the tab page, which displays the resource ID followed by resource name). You can also obtain a resource ID via RAS PowerShell by executing `Get-RASPubItem "resource-name"`. The returned `PubItem` object has the `Id` property that specifies the resource ID. To get the list of all available published resources, execute the `Get-RASPubItem` cmdlet with no parameters. If this key is not set, the Parallels Client will show the list of all available published resources.
Startup	OverrideParams	[Optional] URL encoded arguments to be passed to the published application. If included, the arguments will override existing arguments (if any). The parameter is available in Parallels Clients for all platforms (desktop and mobile) since RAS v16.5.2.

XML Example

The XML document must begin with UTF-8 BOM (byte order mark), which consists of the following 3 bytes:

```
0xEF 0xBB 0xBF
```

Note that BOM is not a text. You need to use a script that can add these 3 bytes to the beginning of the file.

The following is a sample 2XA XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<App xmlns:dt="urn:schemas-microsoft-com:datatypes">
   <IIS>
      <ValidSession dt:dt="ui4">1</ValidSession>
   </IIS>
   <Logon>
      <User>phil@win-rjg64ubucvl</User>
      <LoginEx>win-rjg64ubucvl\phil</LoginEx>
      <Base64ClearPassword dt:dt="bin.base64">SGVsbG9JdHNNZSE=</Base64ClearPassword>
   </Logon>
   <Connection>
      <Port dt:dt="ui4">80</Port>
      <StartMode dt:dt="ui4">0</StartMode>
      <PrimaryServer>192.168.12.73</PrimaryServer>
   </Connection>
   <Startup>
      <PublishedApp dt:dt="string">#1</PublishedApp>
....
      <OverrideParams>Hello%201%202%203</OverrideParams>
   </Startup>
</App>
```

Note that another option to pass the password is to use the `ClearPassword` key instead of `Base64ClearPassword`, as shown below:

```
<ClearPassword>HelloItsMe!</ClearPassword>
```

Log User Off (LogOff)

The **LogOff** command logs off a user from a session on a specified server or from all current sessions on all servers.

Key	Datatype	Value
Command	String	The name of the command to execute. In this instance, it's "LogOff".
Server	String	[Optional] Server name. Server can be specified by IP address or its local name. If this key is omitted, the user will be logged off from all sessions on all servers.
RequestPage	String	[Optional] The path to the current web portal page. This is used in iOS, so that the iOS client can redirect the user back to the web portal page.

URL example

Log off from a specific server:

```
tuxclient:///?Command=LogOff&Server=my.server.testing
```

Log off from all servers:

```
tuxclient:///?Command=LogOff
```

Index

A

Additional Information - 9
API Basics - 5

C

Commands and Options - 13

D

Direct App access - 9

E

Example - 7

G

Get Parallels Client Version (GetVersion) - 13

I

Introduction - 4

J

JSON Payload - 6

L

Launch Published Resources (LaunchApp) - 14
Launch Resources with Advanced Security (Get2xa) - 16
Log User Off (LogOff) - 20

M

Main URL Format - 6

P

Parallels Client URL Scheme - 11

R

RAS Web Client API - 5

U

URL Format - 12
URL Scheme Basics - 11