

Deployment Guide - F5 Integration with RemoteSpark (HTML5 RDP) – DRAFT

Introduction

This document describes the general approach for integration with RemoteSpark HTML5 RDP using Portal Access resource published on APM Webtop. The general approach uses commonly-supported way of passing essential parameters to HTML5 gateways such as user credentials and RDP server address – using URI parameters.

- RemoteSpark:

```
https://<RemoteSpark backend address>/rdpdirect.html?server=%{session.assigned.server}&user=%{session.logon.last.username}&pwd=%{session.sso.token.last.password}&domain=%{session.logon.last.domain}
```

This approach with URI parameters does not look secure, as both user credentials and destination server address are sensitive information and should not be passed to the client.

F5 & RemoteSpark worked closely together to come up with a solution that avoids passing this sensitive information via URI:

- APM passes user credentials to RS GW via HTTP Basic authentication
- APM passes RDP server address to RS GW via special "gw_server" header

This approach provides the following benefits:

- No sensitive information is leaked to the client (like with URI parameters)
- APM can restrict user's access to specific RDP servers. In this integration, RDP server address is specified as a session variable – which is equivalent of RDP resource

About HTML5

Thanks to HTML5 technologies such as Canvas and Websocket, it became possible to create rich applications that run inside a browser and have feature set similar to native applications. Today's HTML5 apps support rendering, audio, printing and file transfer.

Typically, a HTML5 RDP solution would use some rendering protocol on the client (user's browser) and have a server-side component that translates the client-rendering protocol into MS RDP connection to the backend. The client-side part of the solution would communicate with the serverside part typically over WebSocket connection.

Deployment steps

Prerequisites:

- At least one RemoteSpark Servers running on Linux or Windows (more details: <http://www.remotespark.com/html5.html>)
- At least one BIG-IP APM (VE or HW, tested with 11.6.x, 12.x & 13.0)

Spark gateway(s) – Settings:

- Enable Basic authentication in gateway.conf: authorization = Basic • Disable RDPWorker in appcfg.js: useWorker:false

BIG-IP APM – Configuration:

Ensure that basic networking settings are configured:

- VLAN:
 - a. For single-arm configuration - create a VLAN on the interface that will be both accepting connections and used to communicate to RS GW servers.
 - b. For two-arm configuration - create two VLANs, external - for client connections, and internal - for communication with RS GWs.
 - c. Specify VLAN tags if needed, otherwise use untagged
- SelfIP (the IP address used to talk to RS GWs)
 - a. Specify the correct VLAN - should be internal for two-arm config.
 - b. Route (the route BIG-IP will use to reach RS GWs via SelfIP)
 - c. Typically, a 0.0.0.0 route via a router on internal VLAN
 - d. Now, test the configuration by pinging RS GW server via correct VLAN:
ping -I <VLAN name> <RS GW IP>
- LB virtual setup
 - a. Create a pool with RS GW addresses:
 - (i) Go to Local Traffic->Pools, create new.
 - (ii) Specify the IP addresses and ports of RS GW servers.
 - (i) Use monitoring settings as needed. A simple TCP monitor was used in the test setup.
 - (ii) Create a Virtual Server that will be load-balancing RS GW servers:
 - (iii) Go to Local Traffic->Virtual Servers, create new.
 - (iv) Specify destination IP 1.1.1.1 and port 80.
 - (v) Use any other IP/port if desired. Only requirement, from security perspective, is that user should not be able to talk to this IP:port, i.e. it should be internal.
 - (vi) Specify pool that was created in above step. Set Source Address Translation = Automap.

(vii) Now, test that you can reach RS GW by doing `curl http://1.1.1.1` from BIG-IP SSH console. You should get RS GW web page. DO NOT PROCEED IF THIS STEP FAILS!

- Main virtual step
 - 1) Create an SSO profile that will be used to do Basic auth to RS GW and pass RDP server address:
 - 2) Go to Access->Single Sign-On->HTTP Basic, create new.
 - 3) Go to Advanced settings and add a header "gw_server" with value to "%{session.assigned.server}" (or any other APM session variable name that will hold target RDP server address)
 - 4) In Username Source specify "session.domain_user" (or any other APM session variable name that will hold DOMAIN\username value)
 - 5) In Password Source specify "session.domain_pass" (or any other APM session variable name that will hold user's password value)
 - 6) Create a Portal Access resource whose purpose is to publish a link to the Loadbalancing Virtual Server on APM Webtop and activate SSO profile for single sign-on into RS GW
 - 7) Go to Access->Connectivity/VPN->Portal Access, create new.
 - a) Use Patching Type = Full Patching (default)
 - b) Check "Publish on Webtop" checkbox (default)
 - c) Use Link Type = Application URI and specify URI `http://1.1.1.1/rdpdirect.html?server=%{session.assigned.server}`
 - d) Create a Resource Item with following configuration:
 - e) Link Type = Paths
 - f) Destination Type = Hostname, Hostname = *
 - g) Paths = *
 - h) SSO Configuration - specify the SSO profile created above
 - i) Create an Access Policy, assign the Portal Access resource
 - j) Create a webtop object under Access->Webtops. Use Type = Full (default).
 - 8) Go to Access->Policies, create new
 - a) Click Edit Policy - you'll be brought into a new browser tab where you can edit the policy using F5 Visual Policy Editor
 - b) After the Start action, add a Logon Page
 - c) Check the "Split Domain From Username" checkbox in Logon Page configuration
 - d) Next, add AAA agent that will verify user identity.
 - e) Next, add Variable Assign with following variable names / expressions:
 - f) `session.domain_user = expr { "[mcget session.logon.last.domain]\\[mcget session.logon.last.username]" }`
 - g) `session.domain_pass = expr { "[mcget -secure session.logon.last.password]" }`
 - h) `session.assigned.server = expr { "<IP address of RDP backend>" }`
 - 9) Next, add Advanced Resource Assign
 - a) Specify the webtop and Portal Access resource created earlier
 - 10) Finally, create the main Virtual Server
 - 11) Go to Local Traffic->Virtual Servers, create new.
 - a) Specify destination IP address and port 443.

- b) This is the address users will connect with their browsers
- c) Add HTTP, ClientSSL, Rewrite profiles. You can use default ones.
- d) Add Access Profile created one step above.
- e) You'll also be asked to add a Connectivity Profile - click on sign and create one with default values.
- f) Set Source Address Translation = Automap.

Now, test the whole setup by going to <https://<main Virtual Server IP>> from the client device.

Specify DOMAIN\username and password on APM logon page. Click on the Portal Access resource. You should be automatically logged into ORDP server specified in session.assigned.server APM session variable.

Pre-BIG-IP 13.x (-> v11.5+):

Portal Access resources do not support WebSocket connections before v13.x release. This can be fixed with the following iRule that should be assigned on the main Virtual Server:

```
when HTTP_REQUEST { set
tmm_apm_rewrite_request 0
}

when REWRITE_REQUEST_DONE { set
tmm_apm_rewrite_request 1
if { [HTTP::header exists Upgrade] && \
[string tolower [HTTP::header Upgrade]] contains "websocket" } {
# Replace Origin header with backend's address
HTTP::header replace Origin "https://[HTTP::host]"
# Disable rewrite plugin, otherwise request may be stuck
REWRITE::disable
}
}

when HTTP_RESPONSE {
if { !$tmm_apm_rewrite_request } { return }
set tmm_apm_chunked 0
if { [HTTP::header Content-Type] contains "javascript" }
{ if { [HTTP::header Transfer-Encoding] == "chunked" }
{ set tmm_apm_chunked 1
```

```

} else if { [HTTP::header Content-Length] == 0 } { return
}
HTTP::collect 1
}
}

when HTTP_RESPONSE_DATA { if
{ !$tmm_apm_rewrite_request } { return }

set tmm_apm_patch { function
wsWrapper(F5_orig) { var wrapper =
function WebSocket() { if
(typeof(arguments[0]) === 'string')
{ arguments[0] =
F5_WrapURL( arguments[0].replace(/^ws/, '
http'))
.replace(/^http/, 'ws');
}
var socket = new F5_orig(arguments[0], arguments[1]);
socket.constructor = wrapper; return socket;
};
if (Object.setPrototypeOf) {
Object.setPrototypeOf(wrapper, F5_orig);
} else { // IE10
Object.keys(F5_orig).forEach(function (key) { wrapper[key]
= F5_orig[key];
});
}
wrapper.F5_original = F5_orig; wrapper.prototype =
F5_orig.prototype; wrapper.toString = function toString() { return
F5_orig.toString(); }; return wrapper;
};
if ((typeof(window.MozWebSocket) !== 'undefined') &&

```


```

!window.MozWebSocket.F5_original) {
window.MozWebSocket = wsWrapper(window.MozWebSocket)
}
if ((typeof(window.WebSocket) !== 'undefined') &&
!window.WebSocket.F5_original) { window.WebSocket
= wsWrapper(window.WebSocket)
}
}
if { $tmm_apm_chunked } { set tmm_apm_patch "[format %x [string length
$tmm_apm_patch]]\r\n$tmm_apm_patch"
}
HTTP::payload replace 0 0 $tmm_apm_patch
HTTP::release
}
when HTTP_RESPONSE_RELEASE {
if { [HTTP::header exists Content-Security-Policy] } {
HTTP::header replace Content-Security-Policy \
[string map {"data:" "data: blob:"} [HTTP::header Content-Security-Policy]]
}
}
}

```

Screenshots:

Access » Connectivity / VPN : Portal Access : Portal Access Lists » SparkView

 **Properties**

General Properties

Name	SparkView
Partition / Path	Common
Description	<input type="text"/>
ACL Order	<input type="text" value="0"/>

Configuration: Basic ▼

Match Case For Paths	No ▼
Patching	Type Full Patching ▼ <input checked="" type="checkbox"/> HTML Patching <input checked="" type="checkbox"/> JavaScript Patching <input checked="" type="checkbox"/> CSS Patching <input checked="" type="checkbox"/> Flash Patching <input type="checkbox"/> Java Patching
Publish on Webtop	<input checked="" type="checkbox"/> Enable
Link Type	Application URI ▼
Application URI	<input type="text" value="http://1.1.1.1/rdpdirect.html?server=%{session.assigned.server}"/>

Customization Settings for English

Language	English
Caption	<input type="text" value="SparkView"/>
Detailed Description	<input type="text"/>
Image	Choose File No file chosen View/Hide

Update Delete

Resource Items

<input checked="" type="checkbox"/>	Host or IP Address	Port	Paths
<input type="checkbox"/>	*	80	*

Remove



Properties

Resource Item: Advanced ▼

Link Type	Paths ▼
Destination	Type: <input checked="" type="radio"/> Host Name <input type="radio"/> IP Address Host Name *
Paths	*
Scheme	http ▼
Port	80
Headers	Name <input type="text"/> Value <input type="text"/> Add <div></div> Edit Delete

Resource Item Properties: Advanced ▼

Compression	No Compression ▼
Client Cache	Default ▼
SSO Configuration	rs_gw_basic ▼
Session Update	<input checked="" type="checkbox"/> Enabled
Session Timeout	<input checked="" type="checkbox"/> Enabled
Home Tab	<input checked="" type="checkbox"/> Enabled
Log	Packet ▼

Update

Delete

Access » Single Sign-On » rs_gw_basic



Properties

General Properties: Advanced ▼

Name	rs_gw_basic
Partition / Path	Common
SSO Method	HTTP Basic
Headers	<div>Name <input type="text"/></div> <div>Value <input type="text"/></div> <div>Add</div> <div>server: %{session.assigned.server}</div> <div>Edit Delete</div>
Log Settings	+ default-log-setting ▼

Credentials Source

Username Source	session.sso.token.last.username
Password Source	session.sso.token.last.password

SSO Method Configuration

Username Conversion	<input type="checkbox"/> Enable
---------------------	---------------------------------

Update

Delete

Local Traffic » Virtual Servers : Virtual Server List

Virtual Server List

Virtual Address List

Statistics

Search

<input checked="" type="checkbox"/>	Status	Name	Description	Application	Destination
<input type="checkbox"/>		vip_sparkview_443_80			
<input type="checkbox"/>		vip_sparkview_pool			1.1.1.1
<input type="checkbox"/>		vip_sparkview_redirect			

Enable

Disable

Delete...

