volantis® systems

Volantis Mobility Server™ ver. 5.0

Device Repository Web Service

Volantis Mobility Server™

Version 5.0

Device Repository Web Service

Volantis Mobility Server: Device Repository Web Service

Version 5.0
Published date 2008-03-19

Included software

Apache Software Foundation
Davisor Ltd
Free Software Foundation
Jaxen Project
Sun Microsystems Inc

Table of contents

Device Repository Web Service

The Device Repository Web Service (DWS) provides a REST interface over HTTPS to the Volantis Mobile Content Device Repository. An operational example of DWS usage is provided by Storefront 2.0, which populates the user interface with device information, and queries the repository at runtime to match device characteristics to Storefront resource attributes.

You can use either the Eclipse interface to the device update service, or the `mcsUpdateClient` command line utility to download the latest version of an update to a device repository, and the `mcsRepositoryManager` utility to merge repositories and report on the merge.

Code samples that demonstrate the use of several API functions are part of the DWS distribution. To make use of the samples you will need to download these applications. See *Code samples in PHP* and *Code samples in Ruby* for details.

Installing DWS

Installing and configuring the Device Repository Web Service (DWS) is very straitforward, using a wizard that installs all the required files, including a recent repository version, The Tomcat web application server, code samples illustrating the REST API, and command line interfaces to update the repository.

Required software

DWS operates as an application within a web application server. The DWS installation wizard provides an option to install the required Tomcat 5.5.

If you wish to use an existing Tomcat installation you will need to make several settings manually. For example, you might want to use a different certificate. *Configuring the DWS server* outlines the steps required.

Note: DWS is currently supported on Tomcat 5.5 only.

Running the installer

You install DWS with an wizard that steps you through a series of pages containing settings that correspond to the entries in the configuration files. You click **Next** to complete the following actions.

1. Accept the Licence Agreement on the second page
2. Accept the 'Full Install' option for *Install Set*. Alternatively, choose 'Custom' and check the options you require.
3. Enter the install file location for DWS, for example `/workspace/webapps/DRWS`
4. Set the *Username* to 'drws_user' and the *Password* to 'drws_password'
5. On the Installation Summary review your settings
6. Click **Install** to complete the installation

Configuring the installed Tomcat server

If you chose the option to install Tomcat, you will find that Tomcat is installed in the location you specified, with the following predefined configuration.

- The `server.xml` file has a minimal configuration that starts only the SSL connector required for DWS. The original `server.xml` file is also installed, but renamed to `server-default.xml`.
- The `tomcat-users.xml` file contains the user and role required for basic authentication, with the username and password that you entered during installation
- A `.keystore file` with a SSL self-signed certificate generated for host 'localhost'
- A `drws.xml` context file deployed in the `conf/Catalina/localhost` directory

To install DWS as a web application:

1. Create a `drws` directory under `/workspace/webapps/DRWS/tomcat/webapps`
2. Enter the command
   `cp -r /workspace/webapps/DRWS/webapp/* drws`
3. Enter the command
   `export [tomcat_home]=/workspace/webapps/DRWS/tomcat/`
4. In `/workspace/webapps/DRWS/tomcat/bin` start Tomcat by running `startup.sh`
   After setting up SSL the DWS URL will have the following form.

```
https://[username]:[password]@[hostname]:[portnumber]/[context_root]/[request_path]
```

Configuring the DWS server

The DWS installation wizard provides an option to install Tomcat 5.5. If you wish to use an existing Tomcat installation you will need to configure the web application server.

Configuring an existing application server

You configure the DWS servlet by setting the 'repository-url', 'cache-dir', 'cache-refresh-base' and 'cache refresh-frequency' `init-param` element parameter values in the `web.xml` file.

The web service uses basic declarative authentication, so the whole application can be accessed by users with the roles specified in the `web.xml` file.

```
<security-constraint>
   <web-resource-collection>
      <web-resource-name>Entire Application</web-resource-name>
      <url-pattern>/*</url-pattern>
   </web-resource-collection>
   <auth-constraint>
      <role-name>dws-admin</role-name>
   </auth-constraint>
</security-constraint>

<login-config>
   <auth-method>BASIC</auth-method>
   <realm-name>Device Repository Web Service</realm-name>
</login-config>

<security-role>
   <description>The user role required to log in</description>
   <role-name>dws-client</role-name>
</security-role>
```

Roles are defined along with users in the web application user configuration file, for example the `tomcat-users.xml` file, where you can define the roles and associate them with multiple users. The example defines a Tomcat user who has a DWS client role. You should refer to your application server documentation for specific user configuration details.

```
<tomcat-users>
   ...
   <role rolename="dws-client"/><br/>
   <user username="dws_user" password="dws_user" roles="dws-client"/>
   ...
</tomcat-users>
```

The content of the responses is encrypted using HTTPS, so you need to configure the web application server accordingly. For example with Tomcat, proceed as outlined below, following the recommendations in *SSL configuration how-to*.

1. Create a certificate keystore with the command
   ```
   [tomcat_home]/bin/keytool -genkey -alias tomcat
                     -keyalg RSA
   ```
2. At the prompts, complete the details required for the certificate
3. Uncomment and modify or add the SSL `Connector` element attributes in the `[tomcat_home]/conf/server.xml` file to match those for your certificate
4. Restart Tomcat and test the service using `https://localhost:8443`

After setting up SSL the DWS URL will have the following form.

```
https://[username]:[password]@[hostname]:[portnumber]/[context_root]/[request_path]
```

Device logging

In the `drws-config.xml` file,you can use the `logging` element to configure a service to log information from devices that are not recorded in the device repository, and to generate email reports to your device service provider. If the `logging` element is not present, a default log file name is used.

The information includes

- Device name
- Device type
- Additional information from the request such as TAC.FAC or IMEI number, UA profile URL and the header names and values

The `log-file` element specifies the full path to the log file, and the `e-mail-sending` element enables and disables the reporting service. If reporting is disabled, logging will continue.

If email is enabled you use the `config` element container to define the parameters for the SMTP mail server, the email address, subject and frequency information for the reports.

A report is considered to be successfully sent if the email process does not throw exceptions, and the log file is deleted. The process doesn't check if the recipient received and processed the report. If reporting fails, the log entries are not deleted, and the service will retry after a period.

Duplicated entries are filtered out during logging, and the data structure of the filter process is rebuilt from the log file after restarting MCS or DWS. However due to memory constraints duplicated entries are still possible. Duplicates are also possible between different reports.

Details of the configuration elements can be found in *MCS configuration elements*.

DWS API reference

The Device Repository Web Service (DWS) provides a language independent REST interface to the device repository. Requests over HTTPS return listings of device names, policy categories (including the custom category), and policy names and values in an XML format.

The namespace declaration for the web service XML schema is `http://www.volantis.com/xmlns/2006/01/device-ws`.

The table summarizes the request interface.

Request	Description
device-names	Lists the names of all the device policy categories
device-root	Gets the name of the root device in the hierarchy
device-relationship	Returns the relationships between a named device and its immediate surrounding devices in the standard device hierarchy
device-name-by-headers	Identifies the device from the headers
device-name-by-imei	Identifies the device from the 15 digit International Mobile Equipment Identity (IMEI)
device-name-by-tac	Identifies the device from the six digit Type Allocation Code (TAC)
device-name-by-tacfac	Identifies the device from the eight digit combination of Type Allocation and Final Assembly codes (TAC FAC)
device-name-by-ua-profile-url	Identifies the device from the eight digit combination of Type Allocation and Final Assembly codes (TAC FAC)
category-names	Lists the names of all the device policy categories
category-policy-names	Lists all the policies in the specified category
category-info	Returns the category name in a natural language. A *single* language may be specified in the Accept-Language header
policy-names	Lists all the policy names for a device
policy-all-values	Lists all the policy values for a device
policy-value	Returns a specified policy value for a device
policy-values	Returns the specified policy values for a device
policy-info	Returns a policy name and description in a natural language. A *single* language may be specified in the Accept-Language header

Code samples in PHP

When you install DWS, there is an option to install PHP code samples that illustrate the use of the REST API. In the API reference some function descriptions contain links to sample pages.

The PHP samples require PHP 5 and the `HTTP_Request` and `Net_URL` modules from PHP Extension and Application Repository (PEAR)

You can download PHP from:

```
http://www.php.net/downloads.php
```

PHP 5 is configured to install and use the PEAR Package Manager by default, so you can use the Manager's CLI installer with the the following commands.

```
$ pear install HTTP_Request
$ pear install Net_URL
```

You should now be ready to execute the PHP examples. Note that the sample `drws-config.php` is also required to supply server configuraton values.

example1.php

```
<?php
   #
   # (c) Copyright Volantis Systems Ltd. 2006. All rights reserved
   #
   # Simple READ query against a device name
   #

   require_once "HTTP/Request.php";
   require_once "drws-config.php";

   # Web Service base URL
   $base_url = "https://".WS_HOST.":".WS_PORT.WS_PATH;

   # Prepare the GET request with basic authentication
   $req =& new HTTP_Request($base_url."/device/relationships/Nokia-6210");
   $req->setMethod(HTTP_REQUEST_METHOD_GET);
   $req->setBasicAuth(WS_AUTH_USER, WS_AUTH_PASS);

   # Execute the request and print the response
   $result = $req->sendRequest();
   if (PEAR::isError($result)) {
      echo "Error: ".$result->getMessage();
   }
   else {
      echo $req->getResponseBody();
   }
?>
```

example2.php

```
#!/usr/bin/php

<?php
#
# (c) Copyright Volantis Systems Ltd. 2006. All rights reserved
#
# Simple READ query against a set of headers,
# i.e. the device MCS would infer from the supplied headers
#

require_once "HTTP/Request.php";
require_once "drws-config.php";

# Web Service base URL
$base_url = "https://".WS_HOST.":".WS_PORT.WS_PATH;

# Prepare the POST request with basic authentication
$req =& new HTTP_Request($base_url."/device/name-by-headers");
$req->setMethod(HTTP_REQUEST_METHOD_POST);
$req->setBasicAuth(WS_AUTH_USER, WS_AUTH_PASS);
$req->setBody("User-Agent: Nokia 6210");

# Execute the request and print the response
$req->sendRequest();
echo $req->getResponseBody();
?>
```

example3.php

```
#!/usr/bin/php

<?php
#
# (c) Copyright Volantis Systems Ltd. 2006. All rights reserved
#
# Simple READ query against an IMEI
#

require_once "HTTP/Request.php";
require_once "drws-config.php";

# Web Service base URL
$base_url = "https://".WS_HOST.":".WS_PORT.WS_PATH;

# Prepare the GET request with basic authentication
$req =& new HTTP_Request($base_url."/device/name-by-imei/350612201111111");
$req->setMethod(HTTP_REQUEST_METHOD_GET);
$req->setBasicAuth(WS_AUTH_USER, WS_AUTH_PASS);

# Execute the request and print the response
$req->sendRequest();
echo $req->getResponseBody();
?>
```

example4.php

```
#!/usr/bin/php

<?php
#
# (c) Copyright Volantis Systems Ltd. 2006. All rights reserved
#
# Simple READ query against a TAC
#

require_once "HTTP/Request.php";
require_once "drws-config.php";

# Web Service base URL
$base_url = "https://".WS_HOST.":".WS_PORT.WS_PATH;

# Prepare the GET request with basic authentication
$req =& new HTTP_Request($base_url."/device/name-by-tac/35061220");
$req->setMethod(HTTP_REQUEST_METHOD_GET);
$req->setBasicAuth(WS_AUTH_USER, WS_AUTH_PASS);

# Execute the request and print the response
$req->sendRequest();
echo $req->getResponseBody();
?>
```

example5.php

```
#!/usr/bin/php

<?php
#
# (c) Copyright Volantis Systems Ltd. 2006. All rights reserved
#
# Simple READ query against a TACFAC
#

require_once "HTTP/Request.php";
require_once "drws-config.php";

# Web Service base URL
$base_url = "https://".WS_HOST.":".WS_PORT.WS_PATH;

# Prepare the GET request with basic authentication
$req =& new HTTP_Request($base_url."/device/name-by-tacfac/35061235");
$req->setMethod(HTTP_REQUEST_METHOD_GET);
$req->setBasicAuth(WS_AUTH_USER, WS_AUTH_PASS);

# Execute the request and print the response
$req->sendRequest();
echo $req->getResponseBody();
?>
```

example6.php

```
#!/usr/bin/php

<?php
#
# (c) Copyright Volantis Systems Ltd. 2006. All rights reserved
#
# Simple READ query against a UAProfile (URL)
#

require_once "HTTP/Request.php";
require_once "drws-config.php";

# Web Service base URL
$base_url = "https://".WS_HOST.":".WS_PORT.WS_PATH;

# Prepare the POST request with basic authentication
$req =& new HTTP_Request($base_url."/device/name-by-ua-profile-url");
$req->setMethod(HTTP_REQUEST_METHOD_POST);
$req->setBasicAuth(WS_AUTH_USER, WS_AUTH_PASS);
$req->setBody("http://device.sprintpcs.com/Handspring/HSTR300HK");

# Execute the request and print the response
$req->sendRequest();
echo $req->getResponseBody();
?>
```

example7.php

```
#!/usr/bin/php

<?php
#
# (c) Copyright Volantis Systems Ltd. 2006. All rights reserved
#
# Simple READ query retrieving a list of the policies available for a device
#

require_once "HTTP/Request.php";
require_once "drws-config.php";

# Web Service base URL
$base_url = "https://".WS_HOST.":".WS_PORT.WS_PATH;

# Prepare the GET request with basic authentication
$req =& new HTTP_Request($base_url."/device/policy/all-values/Nokia-3410");
$req->setMethod(HTTP_REQUEST_METHOD_GET);
$req->setBasicAuth(WS_AUTH_USER, WS_AUTH_PASS);

# Execute the request and print the response
$req->sendRequest();
echo $req->getResponseBody();
?>
```

example8.php

```
#!/usr/bin/php

<?php
#
# (c) Copyright Volantis Systems Ltd. 2006. All rights reserved
#
# Simple READ query retrieving a list of the policies available in
# the supplied category
#

require_once "HTTP/Request.php";
require_once "drws-config.php";

# Web Service base URL
$base_url = "https://".WS_HOST.":".WS_PORT.WS_PATH;

# Prepare the GET request with basic authentication
$req =& new HTTP_Request($base_url."/category/policy/names/protocol");
$req->setMethod(HTTP_REQUEST_METHOD_GET);
$req->setBasicAuth(WS_AUTH_USER, WS_AUTH_PASS);

# Execute the request and print the response
$req->sendRequest();
echo $req->getResponseBody();
?>
```

example9.php

```
#!/usr/bin/php

<?php
#
# (c) Copyright Volantis Systems Ltd. 2006. All rights reserved
#
# Simple READ query retrieving a list of the policies available in
# the supplied category
#

require_once "HTTP/Request.php";
require_once "drws-config.php";

# Web Service base URL
$base_url = "https://".WS_HOST.":".WS_PORT.WS_PATH;

# Prepare the GET request with basic authentication
$req =& new HTTP_Request($base_url."/category/policy/names/protocol");
$req->setMethod(HTTP_REQUEST_METHOD_GET);
$req->setBasicAuth(WS_AUTH_USER, WS_AUTH_PASS);

# Execute the request and print the response
$req->sendRequest();
echo $req->getResponseBody();
?>
```

Code samples in Ruby

When you install DWS, there is an option to install Ruby code samples that illustrate the use of the REST API. In the API reference some function descriptions contain links to sample pages.

The Ruby samples require Ruby 1.8.2 and the `net/https` module. Note that the sample `drws-config.rb` is also required to supply server configuraton values.

You can download Ruby from:

`http://www.ruby-lang.org/en/`

example1.rb

```
#!/usr/bin/ruby -w

# (c) Copyright Volantis Systems Ltd. 2005. All rights reserved
#
# Simple READ query against a device name
#

require 'net/https'
require 'drws-config'

# Prepare HTTPS connection to the web service
http = Net::HTTP.new(WS_HOST, WS_PORT)
http.use_ssl = true

# Prepare the GET request with basic authentication
req = Net::HTTP::Get.new(WS_PATH + '/device/relationships/Nokia-6210')
req.basic_auth(WS_AUTH_USER, WS_AUTH_PASS)

# Start the connection, execute the request and print the response
# The connection is automatically closed at the end of block
http.start { |http|
    response = http.request(req)
    print response.body
}
```

example2.rb

```
#!/usr/bin/ruby -w

# (c) Copyright Volantis Systems Ltd. 2005. All rights reserved
#
# Simple READ query against a set of headers,
# i.e. the device MCS would infer from the supplied headers
#

require 'net/https'
require 'drws-config'

# Prepare HTTPS connection to the web service
http = Net::HTTP.new(WS_HOST, WS_PORT)
http.use_ssl = true

# Prepare the POST request with basic authentication
req = Net::HTTP::Post.new(WS_PATH + '/device/name-by-headers')
req.basic_auth(WS_AUTH_USER, WS_AUTH_PASS)
req['Content-Type'] = 'text/html'

# Start the connection, execute the request and print the response
# The connection is automatically closed at the end of block
http.start { |http|
    response = http.request(req, 'User-Agent: Nokia 6210')
    print response.body
}
```

example3.rb

```
#!/usr/bin/ruby -w

# (c) Copyright Volantis Systems Ltd. 2005. All rights reserved
#
# Simple READ query against an IMEI
#

require 'net/https'
require 'drws-config'

# Prepare HTTPS connection to the web service
http = Net::HTTP.new(WS_HOST, WS_PORT)
http.use_ssl = true

# Prepare the GET request with basic authentication
req = Net::HTTP::Get.new(WS_PATH + '/device/name-by-imei/350612201111111')
req.basic_auth(WS_AUTH_USER, WS_AUTH_PASS)

# Start the connection, execute the request and print the response
# The connection is automatically closed at the end of block
http.start { |http|
    response = http.request(req)
    print response.body
}
```

example4.rb

```
#!/usr/bin/ruby -w

# (c) Copyright Volantis Systems Ltd. 2005. All rights reserved
#
# Simple READ query against a TAC
#

require 'net/https'
require 'drws-config'

# Prepare HTTPS connection to the web service
http = Net::HTTP.new(WS_HOST, WS_PORT)
http.use_ssl = true

# Prepare the GET request with basic authentication
req = Net::HTTP::Get.new(WS_PATH + '/device/name-by-tac/35061220')
req.basic_auth(WS_AUTH_USER, WS_AUTH_PASS)

# Start the connection, execute the request and print the response
# The connection is automatically closed at the end of block
http.start { |http|
    response = http.request(req)
    print response.body
}
```

example5.rb

```
#!/usr/bin/ruby -w

# (c) Copyright Volantis Systems Ltd. 2005. All rights reserved
#
# Simple READ query against a TACFAC
#

require 'net/https'
require 'drws-config'

# Prepare HTTPS connection to the web service
http = Net::HTTP.new(WS_HOST, WS_PORT)
http.use_ssl = true

# Prepare the GET request with basic authentication
req = Net::HTTP::Get.new(WS_PATH + '/device/name-by-tacfac/35061235')
req.basic_auth(WS_AUTH_USER, WS_AUTH_PASS)

# Start the connection, execute the request and print the response
# The connection is automatically closed at the end of block
http.start { |http|
    response = http.request(req)
    print response.body
}
```

example6.rb

```
#!/usr/bin/ruby -w

# (c) Copyright Volantis Systems Ltd. 2005. All rights reserved
#
# Simple READ query against a UAProfile (URL)
#

require 'net/https'
require 'drws-config'

# Prepare HTTPS connection to the web service
http = Net::HTTP.new(WS_HOST, WS_PORT)
http.use_ssl = true

# Prepare the POST request with basic authentication
req = Net::HTTP::Post.new(WS_PATH + '/device/name-by-ua-profile-url')
req.basic_auth(WS_AUTH_USER, WS_AUTH_PASS)
req['Content-Type'] = 'text/html'

# Start the connection, execute the request and print the response
# The connection is automatically closed at the end of block
http.start { |http|
    response = http.request(req, 'http://device.sprintpcs.com/Handspring/HSTR300HK')
    print response.body
}
```

example7.rb

```
#!/usr/bin/ruby -w

# (c) Copyright Volantis Systems Ltd. 2005. All rights reserved
#
# Simple READ query retrieving a list of the policies available for a device
#

require 'net/https'
require 'drws-config'

# Prepare HTTPS connection to the web service
http = Net::HTTP.new(WS_HOST, WS_PORT)
http.use_ssl = true

# Prepare the GET request with basic authentication
req = Net::HTTP::Get.new(WS_PATH + '/device/policy/all-values/Nokia-3410')
req.basic_auth(WS_AUTH_USER, WS_AUTH_PASS)

# Start the connection, execute the request and print the response
# The connection is automatically closed at the end of block
http.start { |http|
    response = http.request(req)
    print response.body
}
```

example8.rb

```
#!/usr/bin/ruby -w

# (c) Copyright Volantis Systems Ltd. 2005. All rights reserved
#
# Simple READ query retrieving a list of the policies available in the supplied category
#

require 'net/https'
require 'drws-config'

# Prepare HTTPS connection to the web service
http = Net::HTTP.new(WS_HOST, WS_PORT)
http.use_ssl = true

# Prepare the GET request with basic authentication
req = Net::HTTP::Get.new(WS_PATH + '/category/policy/names/protocol')
req.basic_auth(WS_AUTH_USER, WS_AUTH_PASS)

# Start the connection, execute the request and print the response
# The connection is automatically closed at the end of block
http.start { |http|
    response = http.request(req)
    print response.body
}
```

example9.rb

```
#!/usr/bin/ruby -w

# (c) Copyright Volantis Systems Ltd. 2005. All rights reserved
#
# Simple READ query retrieving a policy descriptor
#

require 'net/https'
require 'drws-config'

# Prepare HTTPS connection to the web service
http = Net::HTTP.new(WS_HOST, WS_PORT)
http.use_ssl = true

# Prepare the GET request with basic authentication
req = Net::HTTP::Get.new(WS_PATH + '/policy/info/microphone')
req.basic_auth(WS_AUTH_USER, WS_AUTH_PASS)

# Start the connection, execute the request and print the response
# The connection is automatically closed at the end of block
http.start { |http|
    response = http.request(req)
    print response.body
}
```

References

Apache Software Foundation
    `http://apache.org`
Davisor Ltd
    `http://www.davisor.com`
Free Software Foundation
    `http://www.fsf.org`
Jaxen Project
    `http://jaxen.org/`
Sun Microsystems Inc
    `http://www.sun.com`
SSL configuration how-to
    `http://tomcat.apache.org/tomcat-5.0-doc/ssl-howto`