XNATRestClient

The XNATRestClient is deprecated. You should use XnatDataClient instead for on-going and future development.

XNATRestClient is a Java command-line tool that provides an easy way to script calls to the XNAT REST API. In many ways, **XNATRestClient** is similar to **curl**, although the syntax and usage is somewhat different. This tool is provided for platforms where **curl** may not be installed by default.e

Usage

XNATRestClient is provided as a bash shell script and a Windows batch file (where it's named **XNATRestClient.bat**) and is located in the **bin** folder of your **xnat_builder**. To configure **XNATRestClient** to work properly, all you should need to do is add that **bin** folder to your path variable or specify the path explicitly when you invoke the command.

Parameters

XNATRestClient has the following parameters:

Parameter	Description
-u *username*	The XNAT system username to be used when validating your session.
-p *password*	The password to be used when validating your session.
-user_session *sessionId*	This replaces the -u and -p parameters. You can generate the session ID by calling the REST API URL **/data/JSESSION**.
-host *url*	This is the URL for your XNAT server instance. Note that this includes any application context folder, i.e. **/xnat**.
-ts *store*	Indicates the location of your trusted certificates store.
-tsPass *pass*	The password for your trusted certificates store.
-proxy *url*	The address of your proxy server, if necessary.
-proxyPort *port*	The port to use on your proxy server (the default is port 80).
-h	Print help.
-quiet	Suppresses output messages.
-remote *path*	The remote path to access on your XNAT server. This is relative to the value you specified for the **-host** parameter.
-local *file*	A local file containing data to upload to the REST resource specified by the **-host** and **-remote** parameters.
-m *method*	The HTTP method to use when accessing the specified REST resource.

Common Issues

URL Encoding

XNATRestClient does not encode parameters for URL submission in the value specified in the **-remote** parameter. This can lead to unexpected results when the characters in a REST resource are significant in HTTP transactions. For example, a search specifier might specify a search for data objects with the XSI type of **xnat:demographicsData**. That specifier looks like this:

```
xnat:subjectData/demographics[@xsi:type=xnat:demographicData]/gender=male
```

In this case, the parameter key is *intended* to be **xnat:subjectData/demographics[@xsi:type=xnat:demographicData]/gender** and the parameter value is *intended* to be **male**. But because the key actually already has the equal character ('**=**'), the key actually is interpreted to be **xnat:subjectData /demographics[@xsi:type**. Because of this, you'll need to encode significant characters, including the equal character, ampersands ('**&**') and question marks ('**?**') within your parameter keys and values.

For example, the above specifier would become:

```
xnat:subjectData/demographics[@xsi:type%3dxnat:demographicData]/gender=male
```