IdMUnit Usage Guide

Brent Kynaston & Huston Franklin

IdMUnit Usage Guide

Brent Kynaston & Huston Franklin

Published August 16, 2008

Abstract

This document contains installation and usage instructions for IdMUnit, an automated test framework for Identity Management (IdM) systems. Additional training and documentation is available through IdMUnit training and consulting. Request more information at info@trivir.com

Public Release (GNU GPL)

IdMUnit is a must-have automated testing tool that integrates with Designer for Identity Manager. This suite, based on xUnit and Spring Framework architecture will automatically transform documented test cases into actual automated tests, execute them, and record the results (pass/fail/warning) in text and/or HTML output. This suite provides integrated policy, driver and system testing to accelerate the project release time-frame, ease development risk, control scope, and ensure logical compliance with functional requirements from design until deployment and beyond. Complete functional testing can now be completed for the entire identity management solution in seconds rather than weeks.

IdMUnit provides a quality-control compliment to Identity Manager as it quickly certifies that installed drivers are functioning according to business rules and solution specifications. It also works with TriVir's Data Comparator, which validates the state of every synchronized object and attribute to ensure that synchronization is working as expected throughout connected production systems. TriVir is is working on the release of a freely-distributable, open-source version of IdMUnit. If interested in participating or receiving a demo of the software, please send a request to info@trivir.com to obtain more information.

This document contains instructions for the installation and basic use of IdMUnit. More information can be obtained by emailing info@trivir.com or going to www.IdMUnit.org.

Note: Today the regular development trunk of IdMUnit is maintained within the TriVir LLC corporate headquarters. The open source distribution shares the idmunit-core and is kept up to date with change made at TriVir as needed. Information about IdMUnit or the open source project may be found at www.idmunit.org and http://sourceforge.net/projects/idmunit/

Chapter 1. Installation

1. Installing IdMUnit

1.1. Requirements

- Any operating system supported by Designer for Novell Identity Manager (command-line version of IdMUnit only requires the Java Runtime Environment of 1.5 or higher)
- At least 1024Mb of memory and a 1GHz CPU.
- 40Mb of free disk space

IdMUnit has been tested with:

- Sun Java™ runtime 1.5+ (up to 1.6.0_07) under Windows 2000, XP, Vista, openSUSE Linux 10.2, SUSE Linux Enterprise Desktop 10.x and SUSE Linux Enterprise Server.
- Command line version tested on multiple revisions of Solaris and NetWare in addition to the platforms listed above.
- Designer versions 1.2, 2.1.1 and 3.0.x (including multiple iteration release candidates)

1.2. Install on Linux

Tip

Installing into a virtual machine will provide maximum portability and the ability to take configuration snapshots at desired intervals

1. Download latest release of Designer from here , extract and install by clicking the "install" script in the designer_install directory or invoke it at the command line:

   ```
   $ designer_install/install
   ```

2. Execute any platform-specific recommendations as indicated in the Designer README displayed after installation. For OpenSUSE the following tasks are recommended:

   a. Remove the following lines from the Designer.ini file, which is found in the directory where you installed Designer:

   -XX:+UseParallelGC

   -XX:ParallelGCThreads=20

   -XX:+UseParallelOldGC

   b. Install the libpng3 package using the Software Management component of YaST.

Tip

Use df -h to check the disk space on each partition - select one that has plenty of room to grow

Tip

Installing into a virtual machine will provide maximum portability and the ability to take configuration snapshots at desired intervals

Launch Designer and validate that the installation was successful.

3. Import the current Novell Identity Manager (IDM) project from the target eDirectory tree into Designer. See the Designer help for instructions as necessary.

   Close designer after importing the project.

4. Download the desired Eclipse overlay (with at least the Java development plug-ins). The tested overlay may be download from here:

   Click on the "Eclipse IDE for Java Developers" link on this page: Eclipse Europa

   ### Tip

   The Europa release tends to package all of the necessary related dependencies for Eclipse and the additional packages required by Designer

5. Apply the Eclipse overlay on top of Designer

   ### Tip

   Backing up the config.ini found in Designer/eclipse/configuration and restoring it after the overlay will maintain the Designer branding and packaging

   Copy the folders found in ECLIPSE EXTRACT/eclipse (i.e. configuration, features, plugins and readme) to the Designer/eclipse folder, overwriting all files.

   ### Tip

   Restore the Designer/eclipse/configuration/config.ini file to regain the Designer look & feel
   Restart Designer and validate that project can open.

   If the overlay was successful, the Java perspective will be available (Window --> Open Perspective --> Other --> Java). Switch to the Java perspective now and re-organize windows as necessary. Note that any missing Designer views may be re-opened from Window --> Open view.

1.3. Installing IdMUnit components

Tip

Download the latest release of IdMUnit from http://www.idmunit.org.

1. Creating a New Project of IdMUnit Tests

   ### Tip

   Test projects are now separate from the IdMUnit core project in IdMUnit 2.0 and above.

   1. Create a new test project folder (i.e. ABCTests)
   2. Copy the profile files from the IdMUnit examples folder found in the distribution (~examples/profiles/*) into the new folder ABCTests
   3. Copy the org container from the examples found in the distribution (~examples/tests/*) into the new ABCTests
   4. Update the spreadsheet names and their corresponding test runners in the newly copied IdMUnit folder (ABCTests/org/idmunit/) - See "Creating a New Test Spreadsheet and Runner for more details.
   5. In Designer ensure that the Java perspective is opened and create a new Java project following these steps:

- Click File --> New --> Java Project --> Add a name at the top then select the "Create project from existing source" bullet and browse to the newly created test project directory (i.e. ABCTests) --> Finish
- Add the IdMUnit libraries to the test project by following the steps described in "Using IdMUnit Binary Distribution" or "Using IdMUnit Source" below.

2. Importing an Existing IdMUnit Test Project
    1. If a test project has already been created (see "Creating a New Project of IdMUnit Tests") it may be imported into a Designer workspace by following these steps:
        - Click File --> Import --> General --> Existing Projects into Workspace --> Next
        - Browse to the project of tests that should be imported (i.e. ABCTests) and click Finish
3. Using IdMUnit Binary Distribution
    1. Copy idmunit.jar, from dist, and all of the JAR files from lib to a lib subdirectory in your test project.
    2. Select Project#Properties to open the Properties dialog.
    3. Select Java Build Path from the tree control on the left.
    4. Select the Libraries tab and click Add JARs...to open the JAR Selection dialog.
    5. Browse to the JARs you copied into your project, select all of them, and click OK.
    6. Click OK to close the project Properties dialog.
4. Using IdMUnit Source
    1. Download the latest IdMUnit source from http://sourceforge.net/projects/idmunit/
    2. In Designer (Java perspective) select File # Import # Existing projects into workspace
    3. Click the Browse button and browse to the idmunig-core folder that was just downloaded (this is the IdMUnit engine source directory)
    4. The IdMUnit should appear in the box as selected. Click Finish.
5. Creating a New Test Spreadsheet and Runner
    1. You are now ready to create the Excel workbook and the corresponding Java class for your project. These should be created in a sub-directory, typically org\idmunit, of the src directory in your project.

        ## Tip

        All standard supported IdMUnit verbs are found in idmunit-core/docs/IdMUnitReference.xls

        The Excel workbook name must consist of only alpa-numeric characters and must not start with a number.

        The name of the Java class must match the name of the workbook.

        The Java class must extend the org.idmunit.IdMUnitTestCase class.

        Any sheets you want to execute from the workbook need a corresponding method in the Java class.

        The method should consist of only a call to the executeTest method.

        The Java class must include a suite method with a call to DDStepsSuiteFactory.createSuite.

        ```
        public static Test suite() {
        ```

```
    return DDStepsSuiteFactory.createSuite(YourClass.class);
}
```

6. Basic Test Execution

   Test configurations may be executed as follows:

   1. Expand the test project in the Package Explorer view of Designer (with Eclipse overlay)
   2. Open the desired test runner in Designer (this file will be named with the same name as the target test spreadsheet but with a .Java file extension) by double-clicking the file name.
   3. Enable or disable the desired test sheets that should be executed from the spreadsheet/workbook. Double forward-slash will de-activate a line.
   4. Execute the spreadsheet tests by right clicking the window (or the file name in the Package Explorer) and run as JUnit test.
   5. Monitor progress in the JUnit plug-in of Designer.

      ### Tip

      After updating test configuration in a particular spreadsheet it should be saved and then refreshed in Designer to pull the latest test configuration into the project. A spreadsheet can be refreshed in Designer by right-clicking and selecting refresh or by hitting F5 while selected.

2. IdMUnit Spreadsheet Content: Identity Management-related Nouns and Verbs

2.1. Spreadsheet usege vocabulary

- IdMUnit talks directly to each IDM-connected system to generate and validate transactions and data. The transactions generated by IdMUnit include all those possible and common in identity management integrations such as object add, modify, delete, rename, move, enable, disable, password initialization, password sync, locked by intruder, etc.
- Refer to the IdMUnit 2.0 Reference Spreadsheet for examples on all officially supported transactions and validations (found in idmunit-core/docs/IdMUnitReference.xls

Chapter 2. Configuration

1. Configuring IdMUnit

1.1. Profile Configuration

- Profiles contain a collection of connection information required for target systems. A single profile represents all of the necessary information required to talk to Identity Manager-connected systems within a single environment, like the development lab, test lab, or production environment. For additional details on the layout of a profile, see IdMUnit.dtd.

  Sample profile excerpt:

```
<profile name="trivirLab">
    <connection>
      <name>META</name>
      <description>Connector for the identity vault</description>
      <type>org.IdMUnit.connector.LDAP</type>
      <server>172.19.18.132</server>
      <user>cn=admin,o=services</user>
      <password>B2vPD2UsfKc=</password>
      <keystore-path/>
```

1.2. Profile Selection

- The “live-profile” specified at the top of the IdMUnit-config.xml file will be the profile used by IdMUnit for test execution.

  For example:

```
<IdMUnit live-profile="trivirLab" enable-email-alerts="true" enable-log-alerts="true">
```

  This configuration would execute tests against the connections specified in the “trivirLab” profile. It would also write a log and send email alerts if any tests fail that have been marked as "IsCritical" in the spreadsheet.

- Profile Use in a Multi-User Environment

  Available upon request in the IdMUnit 2.0 Training Session (more info available at info@trivir.com)

1.3. Password Encryption for Use in the IdMUnit Profile Configuration

- Passwords need to be encrypted using DES and base-64 encoded in order to be used in idmunit-profile.xml. The following steps will encrypte and encode a password:

1. Open the EncTool class in the Package Explorer
2. Expand the IdMUnit-core folder in the Package Explorer --> expand the org.IdMUnit package --> right-click EncTool and select Run-AS --> then select the green Run button
3. Select Java application in the left pane of the new pop-up window
4. Click the “new” icon in the top left-hand corner and a configuration layout will appear to the right
5. Select the arguments tab

6. In the Program Arguments field, type the following where mypassword is the actual password of the system like so:

7. `IDMUNIT1 mypassword`

8. Click Run at the bottom and the password will be encrypted with DES and base-64 encoded.

9. Copy the generated password to the IdMUnit-profiles.xml for the target system

10. Note that the key IDMUNIT1 may be changed, as long as the bytes for the key are specified in the top of the EncTool source in the “iv” class member variable:

```
private final byte[] iv = { 0x0a, 0x01, 0x02, 0x03, 0x04, 0x0b, 0x0c, 0x0d };
```

1.4. Import a Certificate into IdMUnit for SSL Communication via the LDAP Connector

- This procedure follows the steps used to configure a standard Java Keystore:

  1. Export a DER encoded trusted-root certificate (from the target AD DC, eDirectory or other LDAP server)

  2. Copy the DER encoded certificate onto the machine where IdMUnit is running. It may be copied into IdMUnit/util/security where a batch file resides that will help with importing the certificate into a Java certificate store

  3. Edit the IdMUnit/util/security/importCert.bat file to import the certificate into a known keystore as follows where MYCERT.cer is the name of the exported certificate file and the MYSSLKEY is the alias name of this certificate in the keystore:

  4. SAMPLE importCert.bat:

     ```
     java sun.security.tools.KeyTool -import -file MYCERT.cer -keystore keystore -alias "type=r.name=MYSSL
     ```

  5. When executed, the keytool will ask for a password. Any password may be assigned to the keystore (remember this password in order to add additional certificates later on). The certificate file name and alias name may be parameterized with %1 and %2.

Chapter 3. Tutorial: Command-line IdMUnit Scripting and Scheduling

- Available in the IdMUnit 2.0 Training Session (more info available at info@trivir.com)

Chapter 4. Best Practices: Accelerating Development, Testing and IDM Productivity with IdMUnit

- Available in the IdMUnit 2.0 Training Session (more info available at info@trivir.com)