



www.internet2.edu

DCN Software Suite v0.3: OSCARS Inter-Domain Controller (IDC) Installation Guide

Table of Contents

| | | |
|-------|---|---|
| 1 | Overview | 1 |
| 1.1 | About this Document | 1 |
| 1.2 | Hardware and Software Requirements..... | 1 |
| 1.2.1 | System Requirements..... | 1 |
| 1.2.2 | Network Requirements | 1 |
| 1.2.3 | Firewall Requirements | 1 |
| 1.2.4 | Third-Party Library and Package Requirements..... | 1 |
| 1.3 | Downloading the IDC software..... | 2 |
| 2 | Preparing your Environment..... | 2 |
| 2.1 | MySQL..... | 3 |
| 2.1.1 | Install Option 1: Manual Installation | 3 |
| 2.1.2 | Install Option 2: Automatic Installation with a Package Manager | 3 |
| 2.2 | Java Development Kit (JDK) | 3 |
| 2.2.1 | Do I already have the right version of Java?..... | 4 |
| 2.2.2 | Download and Installation | 4 |
| 2.2.3 | Setting the JAVA_HOME Environment Variable..... | 4 |
| 2.2.4 | Optional: Adding JAVA_HOME/bin To Your PATH Variable | 5 |
| 2.3 | Tomcat..... | 5 |
| 2.3.1 | Download and Installation | 5 |
| 2.3.2 | Setting the CATALINA_HOME Environment Variable | 5 |
| 2.3.3 | Starting/Stopping the Tomcat Server..... | 6 |
| 2.3.4 | Verifying a Successful Installation | 6 |
| 2.3.5 | Configuring SSL | 6 |
| 2.4 | Ant..... | 6 |

| | | |
|-------|---|----|
| 2.4.1 | Download and Installation | 6 |
| 2.4.2 | Setting the ANT_HOME Environment Variable..... | 7 |
| 2.4.3 | Adding ANT_HOME/bin to Your PATH Variable..... | 7 |
| 2.5 | Java Transaction API (JTA) | 7 |
| 2.6 | SMTP Server | 8 |
| 3 | Installing the Inter-Domain Controller (IDC) Software | 8 |
| 3.1 | Installing the IDC for the First Time..... | 8 |
| 3.2 | Upgrading an Existing IDC from Version 0.2 | 11 |
| 3.3 | Installing the TERCE | 13 |
| 3.4 | Configuring OSCARS with MySQL..... | 13 |
| 3.5 | Verifying the Web Service API Installation | 14 |
| 3.6 | Verifying the Web User Interface (WBUI) Installation..... | 15 |
| 4 | Creating and Managing User Accounts | 15 |
| 4.1 | Adding Users to the Database | 15 |
| 4.1.1 | Creating a New User Accounts..... | 15 |
| 4.1.2 | Modifying Users | 16 |
| 4.1.3 | Deleting Users..... | 16 |
| 4.2 | Managing X.509 Certificates | 16 |
| 4.2.1 | Keystores..... | 16 |
| 4.2.2 | Using an External CA | 17 |
| 4.2.3 | Generating Your Own CA | 17 |
| 4.2.4 | Signing User ‘Certificate Signing Requests’ (CSRs) | 18 |
| 5 | Describing Your Network Topology | 18 |
| 5.1 | Generating an XML Topology Description | 18 |
| 5.1.1 | Example XML files..... | 19 |

| | | |
|--------|---|----|
| 5.1.2 | Domains, Nodes, Ports and Links | 19 |
| 5.1.3 | Fully-Qualified Identifiers | 19 |
| 5.1.4 | <topology> Element | 20 |
| 5.1.5 | <domain> Element..... | 20 |
| 5.1.6 | <node> element..... | 20 |
| 5.1.7 | <port> Element | 21 |
| 5.1.8 | <link> Element | 22 |
| 5.1.9 | <switchingCapabilityDescriptors> Element | 22 |
| 5.1.10 | <switchingCapabilitySpecficInfo> Element | 23 |
| 5.2 | Creating a Static List of Local Paths | 23 |
| 5.3 | Configuring the <code>terce-ws.properties</code> file | 24 |
| 5.4 | Populating the Scheduling Database | 25 |
| 5.4.1 | Defining Your Local Domain | 25 |
| 5.4.2 | Run <code>updateTopology.sh</code> | 25 |
| 6 | Preparing for Circuit Creation | 26 |
| 6.1 | Configuring <code>oscars.properties</code> | 26 |
| 6.1.1 | MySQL Database Properties..... | 26 |
| 6.1.2 | Authentication, Authorization, and Accounting (AAA) Properties..... | 26 |
| 6.1.3 | Topology Exchange and Pathfinding Properties..... | 27 |
| 6.1.4 | Path Setup Properties | 28 |
| 6.1.5 | Other Properties | 29 |
| 6.2 | Running the Scheduler | 30 |
| 6.3 | Using Hostnames Instead of URNs in Requests | 30 |
| 6.4 | Verifying that You Can Build Local Circuits | 30 |
| 7 | Inter-Domain Configuration | 31 |

| | | |
|-------|---|----|
| 7.1 | Making your IDC Aware of Other Domains..... | 31 |
| 7.2 | Building Your Inter-Domain Routing Table..... | 32 |
| 7.2.1 | Populating the <remoteLinkId> fields in <code>tedb-intra.xml</code> and <code>tedb-inter.xml</code> | 32 |
| 7.2.2 | Defining a Default Route | 33 |
| 7.2.3 | Defining a Path with Only an Egress | 33 |
| 7.2.4 | Defining a Path with Multiple Hops | 34 |
| 7.3 | Generating a Server Certificate for Inter-Domain Requests | 35 |
| 7.4 | SSL Certificates..... | 36 |
| 8 | Advanced Installation | 36 |
| 8.1 | Axis2 | 36 |
| 8.1.1 | Download and Installation | 36 |
| 8.1.2 | Setting the <code>AXIS2_HOME</code> Environment Variable..... | 36 |
| 8.1.3 | Verifying a Successful Installation | 37 |
| 8.2 | Rampart | 37 |
| 8.2.1 | Download and Installation | 37 |
| 8.2.2 | Verifying a Successful Installation | 37 |

1 Overview

1.1 About this Document

This document is intended to be a guide for installing the OSCARS Inter-Domain Controller (IDC) as part of the DCN Software Suite. It specifically targets those interested in installing an IDC on a network running the DRAGON software (also included in the DCN Software Suite). The document assumes basic familiarity with DRAGON and experience with a Unix-like operating system. It does not assume experience with XML or building Java software but such experience may be useful.

1.2 Hardware and Software Requirements

1.2.1 System Requirements

The OSCARS IDC software requires a single PC that will act as a web server for processing requests. Most modern PCs should be suitable for running the software. The following specifications are the minimum requirements for most installations:

- 1Ghz Processor (preferably X86 architecture), 1GB memory
- Linux/Unix Operating System
- Basic Internet connectivity
- System clock running the Network Time Protocol (NTP)

Requirements may be greater for systems running the OSCARS IDC concurrently on the same machine as other components of the DCN Software Suite.

1.2.2 Network Requirements

A network running the DRAGON control plane software is required for this installation. See the DRAGON documentation for more information. The DRAGON software and documentation location is as described in section 1.3 of this document.

1.2.3 Firewall Requirements

The IDC runs on port 8080 and port 8443 by default.

1.2.4 Third-Party Library and Package Requirements

Installing and running the OSCARS IDC requires the following software packages:

| Name | Supported Version | Download Location |
|------|-------------------|-------------------|
|------|-------------------|-------------------|

| | | |
|---|--------|---|
| MySQL | 4.1+ | http://dev.mysql.com/downloads/mysql/ |
| Java Development Kit (JDK) | 5.0 | http://java.sun.com/javase/downloads/index_jdk5.jsp |
| Tomcat | 5.5 | http://tomcat.apache.org/download-55.cgi |
| Axis2 | 1.3 | http://ws.apache.org/axis2/download/1_3/download.cgi |
| Rampart | 1.3 | http://www.apache.org/dyn/mirrors/mirrors.cgi/ws/rampart/1_3/rampart-1.3.zip |
| Ant | 1.7 | http://ant.apache.org/bindownload.cgi |
| Java Transaction API (JTA)¹ | 1.0.1b | http://sourceforge.net/project/showfiles.php?group_id=40712&package_id=127784&release_id=529023 |

1.3 Downloading the IDC software

The IDC software is part of the DCN Software Suite. It can be downloaded at:

- <https://wiki.internet2.edu/confluence/display/DCNSS>

After downloading the DCN software suite, you may unpack it with the following commands:

```
% gunzip dcn-software-suite-0.3.tar.gz
% tar -xvf dcn-software-suite-0.3.tar
```

This will create a directory called `dcn-software-suite-0.3`. The IDC software, called OSCARS, is located in the subdirectory `dcn-software-suite-0.3/idc`. The TERCE software is located in `dcn-software-suite-0.3/terce`. This document describes the installation of both the IDC and TERCE.

The DRAGON software is located in the subdirectory `dcn-software-suite-0.3/dragon`. DRAGON software installation instructions are located in `dcn-software-suite-0.3/dragon/docs/DRAGON-INSTALL-0.3.pdf`.

The remainder of this document will focus on the IDC installation.

2 Preparing your Environment

This section details how to install and configure prerequisite software on the machine that will be running the IDC. There are 5 steps in this process you must do manually:

1. Install MySQL

¹ The link given is to a library called Hibernate that contains JTA. This is currently the easiest way to obtain the library.

2. Install the Java Development Kit and set the JAVA_HOME environment variable
3. Install the Tomcat web server and set the CATALINA_HOME environment variable
4. Install Ant and add it to your PATH environment variable
5. Install the Java Transaction API (JTA)

In addition the Axis2 and Rampart libraries from Apache must be installed. This will be done automatically by the OSCARS installation script as described in Section 3. If you would like to perform this installation manually please see section **8 Advanced Installation**.

2.1 MySQL

MySQL is the database used to maintain user accounts and track reservations. You may install MySQL in one of two ways: manually, by installing a package downloaded from the MySQL web site OR automatically, using your operating system's package manager:

2.1.1 Install Option 1: Manual Installation

Download the MySQL package from the MySQL web site at:

- <http://dev.mysql.com/downloads/mysql>

Installing MySQL in this manner is beyond the scope of this document but (English) installation instructions may be found at:

- <http://dev.mysql.com/doc>

2.1.2 Install Option 2: Automatic Installation with a Package Manager

Download and install MySQL through a package manager if your operating system runs such a service. A few common package managers are up2date (RedHat), apt-get (Debian), and yum. You may install MySQL using a command such as:

```
% up2date mysql-server
```

Consult specific package managers for the exact command and package name.

2.2 Java Development Kit (JDK)

Java is the programming language in which the OSCARS IDC software was created and provides the environment in which it runs. In addition to running the software, the JDK also contains utilities required for compiling the source code and generating user certificates. This section details installation and configuration related to this package.

2.2.1 Do I already have the right version of Java?

Many systems come pre-installed with Java. To install the IDC, your system must not only have Java Runtime Environment (JRE) version 5 but also the various compilers and utilities. To verify that you have the necessary Java environment, issue the following command:

```
% javac -version
```

If the first line of output reads `javac 1.5.0_X`, you should not need to install the Java Development Kit and may skip to section **2.2.3 Setting the JAVA_HOME Environment Variable**. If you get “command not found” or the version number is less than 1.5, you may need to install JDK 5.0 and should proceed to **2.2.2 Download and Installation**.

NOTE: If you are not running the SUN distribution of Java you may encounter issues. The GNU and IBM versions of Java are not fully tested and some users have reported problems. It is recommended you run the SUN distribution of Java.

2.2.2 Download and Installation

You may download JDK 5.0 from Sun’s web site at:

- http://java.sun.com/javase/downloads/index_jdk5.jsp

It is recommended that you download the latest update of **JDK Version 5**. Choose the package most suitable for your operating system.

Once downloaded, unpack the file; this should create a new folder named something similar to “jdk1.5.0_X”. The final step of installation is to move this folder to an easily accessible place. We recommend renaming the folder to java5 in /usr/local with the following command:

```
% sudo mv jdk1.5.0_X /usr/local/java5
```

The location may be anywhere you choose – just make sure you note the location as it is required for setting the JAVA_HOME environment variable in the next section.

2.2.3 Setting the JAVA_HOME Environment Variable

Once Java is installed, you need to set the JAVA_HOME environment variable with its location. This variable is required by the Tomcat web server (see section **2.3 Tomcat**) to run. To set this environment variable, issue these commands:

```
% JAVA_HOME=/usr/local/java5
% export JAVA_HOME
```

You may permanently set this variable (recommended) by adding the above commands to the profile file in your home directory (i.e. `.bash_profile` or `.profile`).

2.2.4 Optional: Adding JAVA_HOME/bin To Your PATH Variable

This step is optional but may make issuing commands easier in later steps. You should add the folder JAVA_HOME/bin to your PATH environment variable so that you can easily access the `keytool` command for issuing certificates. To update your PATH variable, issue these commands:

```
% PATH=$PATH:$JAVA_HOME/bin
% export PATH
```

You may permanently set this variable (recommended) by adding the above commands to the profile file in your home directory (i.e. `.bash_profile` or `.profile`).

2.3 Tomcat

Tomcat is a Java-based application container in which the OSCARS IDC software runs. This section details installation and basic configuration of Tomcat.

2.3.1 Download and Installation

You may download Tomcat from the project's web site at:

- <http://tomcat.apache.org/download-55.cgi>

It is recommended you download **Tomcat Version 5.5**.

NOTE: Tomcat 6.0 is NOT currently supported by the software.

Once downloaded, unpack the downloaded file; this should create a new folder named something similar to "apache-tomcat-5.5.X". The final step of installation is to move this folder to an easily accessible place. We recommend renaming the folder to tomcat in /usr/local with the following command:

```
% sudo mv apache-tomcat-5.5.X /usr/local/tomcat
```

The location may be anywhere you choose – just make sure you note the location as it is required for setting the CATALINA_HOME environment variable in the next section.

NOTE: It is also possible to download and install Tomcat via a package manager; however, some users have reported difficulty with this method. If you install Tomcat using this method be aware that some of the environment variables and other settings may vary from what is contained within this document.

2.3.2 Setting the CATALINA_HOME Environment Variable

Once Tomcat is installed, you need to set the CATALINA_HOME environment variable with its location. This variable is required by the Tomcat web server to run. To set this environment variable, issue these commands:

```
% CATALINA_HOME=/usr/local/tomcat
% export CATALINA_HOME
```

You may permanently set this variable (recommended) by adding the above commands to the profile file in your home directory (i.e. `.bash_profile` or `.profile`).

2.3.3 Starting/Stopping the Tomcat Server

You may start Tomcat with the following command:

```
% $CATALINA_HOME/bin/startup.sh
```

You shutdown the Tomcat server with the following command:

```
% $CATALINA_HOME/bin/shutdown.sh
```

2.3.4 Verifying a Successful Installation

To verify installation was successful, startup the Tomcat server with the following command:

```
% $CATALINA_HOME/bin/startup.sh
```

After starting the server, point a web browser to port 8080 of the machine on which you installed Tomcat with the following URL:

- <http://your-machine-name:8080>

If installation was successful, a web page will load with the Tomcat logo and a message that reads “If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!”

2.3.5 Configuring SSL

You may configure Tomcat to use SSL so that all requests and responses to the server are encrypted. This is not required but highly recommended. Information on this process can be found at:

- <http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html>

2.4 Ant

Ant is a tool that is used to build the IDC code and deploy various configuration files (think of it as “make” for Java). This section details how to install and configure Ant.

2.4.1 Download and Installation

Download Ant from the project's web site at:

- <http://ant.apache.org/bindownload.cgi>

Most IDC testing has been done with **Version 1.7**. Unpack and install Ant with the following commands:

```
% unzip apache-ant-1.7.0-bin.zip
% sudo mv apache-ant-1.7.0 /usr/local/ant
```

You are not required to install the downloaded folder in /usr/local/ant but you should note where it is installed as this information is needed in later steps.

2.4.2 Setting the ANT_HOME Environment Variable

Once Ant is installed, you need to set the ANT_HOME environment variable with Ant's location. To set this environment variable, issue these commands:

```
% ANT_HOME=/usr/local/ant
% export ANT_HOME
```

You may permanently set this variable (recommended) by adding the above commands to the profile file in your home directory (i.e. .bash_profile or .profile).

2.4.3 Adding ANT_HOME/bin to Your PATH Variable

It is recommended you add the Ant bin directory to your PATH environment variable. This will allow ant command-line tools to be found when you type-in the command name. To set this environment variable, issue these commands:

```
% PATH=$PATH:$ANT_HOME/bin
% export PATH
```

You may permanently set this variable (recommended) by adding the above commands to the profile file in your home directory (i.e. .bash_profile or .profile).

2.5 Java Transaction API (JTA)

The Java Transaction API (JTA) is a library used for managing database connections in the IDC. It is a single JAR file but its license restricts distribution with the DCN Software Suite. The easiest way to obtain this library is to download another package called Hibernate and extract the appropriate JAR. The steps for obtaining and installing JTA are:

1. Download and unpack the DCN Software Suite (see section **1.3 Downloading the IDC software**)
2. Download the Hibernate-Core library at:
 - http://sourceforge.net/project/showfiles.php?group_id=40712&package_id=127784&release_id=529023
3. Unpack the downloaded files with the following command:

```
% unzip hibernate-3.2.5.ga.zip
```

4. Copy the `jta.jar` file to the `lib` directory of downloaded IDC software. The command to do this is:

```
% cp hibernate-3.2/lib/jta.jar dcn-software-suite-0.3/idc/lib
```

2.6 SMTP Server

You may **optionally** install an SMTP server on the same machine as the IDC and it will send email notification of circuit activity. A standard installation of `sendmail` should be suitable for the IDC's purposes. The IDC sends mail by transmitting SMTP packets to `localhost` so it is not dependent on the underlying software. Installing such software is beyond the scope of this document.

3 Installing the Inter-Domain Controller (IDC) Software

This section details how to install the OSCARS IDC Software. It assumes you have installed all the prerequisites as described in the previous section, **Preparing your Environment**. This section will cover installing an IDC for the first time, upgrading from a previous version of the IDC, and installing the TERCE component.

3.1 Installing the IDC for the First Time

This section describes how to install OSCARS on a machine that does not have another version of the software already running. **If you have a previous version of the OSCARS IDC installed that you wish to upgrade, please proceed to the next section of this document.** The steps in this section will install both the OSCARS web page interface for human users and the web service interface for applications. To install OSCARS for the first time run the `do_build.sh` script followed by `do_install.sh`. These two scripts will compile the code, build the necessary MySQL database tables, and copy configuration files to the appropriate locations. Below is a detailed description of their use.

First run `do_build.sh` and you should see the following:

```
$ ./do_build.sh

--- Checking prerequisites...
We seem to be in the correct directory
Found ant
Environment variable CATALINA_HOME is set to
/usr/local/tomcat
Environment variable DOMAIN_HOME is not set. Continuing
without it.
Found jta.jar under lib
Axis2 library not found.
```

```
Rampart library not found.
```

If you do not see the above output verify that you have installed all the prerequisites described in **2 Preparing your Environment**. You should answer ‘y’ to the next two questions and proceed as shown below:

```
- Axis2 installation not detected. Should I build it for you
y/n? y
    OK, will build Axis2 for you.

- Axis2 is not deployed. Should I do this for you y/n? y
    OK, will deploy Axis2 for you.
--- Downloading Axis2...
```

You should then see output of Axis2 and the module Rampart downloading. The script will then restart your Tomcat server. The output for the Axis2/Rampart download and the Tomcat restart is not shown in this document as it is rather lengthy. You should know that if your Tomcat server is not running you may see a harmless “Connection refused” exception. This only displays because the script first tries to shutdown the Tomcat server, which it can only do if Tomcat is running. This exception should not affect you installation so please ignore it if it appears. After restarting Tomcat you should see the following:

```
--- Your kit looks good.
- Input the hostname for this IDC. Leave blank for "your-
server.com":
```

If the host name displayed is not correct for your machine then enter the correct one, otherwise you may just hit the return key. You will then be asked if you would like to setup your MySQL database. You should answer ‘y’ to the questions and provide information for an account capable of creating and granting privileges on databases as shown below:

```
- Install databases y/n? y
    OK, will install databases.
    Found mysql client at /usr/bin/mysql
- Input the MySQL server hostname. Leave blank for localhost:
    Using localhost .
- Input a privileged MySQL username on that host. Leave blank
for root:
    Using root .
- Input the password for the privileged account:
    Privileged account access verified.
```

The next of questions will ask for information on the MySQL user that will be used by the IDC to connect to the database. Please remember the information you specify for later as it will be important when configuring your IDC. Example output is shown below:

```

- Input a MySQL username the IDC will use to connect to the
databases.
  -- This name and password must match the
hibernate.connection.username and password specified in
oscars.properties.
  Leave blank for "oscars":
    Using oscars .
- Input the password for the IDC account:
  IDC account access verified.

- Got all information. Press return to create the databases...
  Creating databases bss, aaa, testbss, testaaa
  Databases created...
  Initializing databases...
  Databases initialized...
  Granting privileges to IDC account...
  IDC account authorized.
  Modifying conf/server/aaa.cfg.xml ...
  Modifying conf/server/bss.cfg.xml ...

```

You are now ready to build your IDC as indicated by the following output:

```

- Press return to build IDC...

```

As the last line indicates pressing return will build (compile) the IDC. You will know the build was successful when you see the following output:

```

BUILD SUCCESSFUL
Total time: 30 seconds
--- IDC built.

```

A final prompt will appear asking if you would like to build the IDC tools. You should answer 'y' to this prompt. It will compile a number of utilities important for configuring and running your IDC. This includes the circuit scheduler and commands for managing the database. Below is a snippet of the output you will see:

```

Should I build the OSCARS tools for you y/n?y
...
--- Tools built.

```

At this point you have successfully built the IDC. The next step is to deploy the software on your system with `do_install.sh`. The script will start by stopping your Tomcat server and prompting you to tell it if it should copy the OSCARS configuration files to their proper locations:

```
% ./do_install.sh
--- Checking prerequisites...
    We seem to be in the correct directory
    CATALINA_HOME is set to /usr/local/tomcat

--- Stopping Tomcat...
Using CATALINA_BASE:  /usr/local/tomcat
Using CATALINA_HOME:  /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:       /usr/local/java5

Do you wish to copy key files and oscars configuration files
to Tomcat now? [y/n] y
```

You should answer ‘y’ and what follows is a long stream of output that moves the OSCARS software and configuration files to their correct locations. You will know installation was successful when you see the following:

```
IDC deployed, server configured
```

You are now ready to install the TERCE component and begin configuring the OSCARS software. Proceed to **3.3 Installing the TERCE** to continue installation.

3.2 Upgrading an Existing IDC from Version 0.2

You may upgrade your IDC using the `do_upgrade.sh` and `do_install.sh` scripts. The `do_upgrade.sh` will automatically make changes to the MySQL database and configuration files necessary for an upgrade from version 0.2 of the software. Users of 0.1 should first upgrade to version 0.2 or perform a clean installation of the software for the best results.

You start your upgrade by running the `do_upgrade.sh` command as follows:

```
% ./do_upgrade.sh
--- Checking prerequisites...
    We seem to be in the correct directory
    CATALINA_HOME is set to /usr/local/tomcat
```

A number of prompts may then appear. **You may see some or all of the listed prompts depending on your particular installation.** The first prompt you will see asks if you would like to upgrade your MySQL tables. You should answer ‘y’ if you have not run this script before. It will create two new tables and add a column to another. It will NOT damage any of your previous data in the database. This prompt is shown below:

```
Would you like to upgrade your MySQL tables y/n? y
Please enter your mysql user name: oscars
--- mysql tables upgraded
```


You may then be asked to copy `sec-server.jks` to a directory in `CATALINA_HOME`. You should also do this so OSCARS can properly authenticate incoming requests. The output looks like the following:

```
May I copy sec-server.jks to
$CATALINA_HOME/shared/classes/server y/n? y
--- sec-server.jks moved to
$CATALINA_HOME/shared/classes/server/
```

If you do not have a `sec-server.properties` file installed then you will be prompted to copy a default version to a directory in `CATALINA_HOME`. OSCARS needs this file so it can properly read `sec-server.jks`. Your output will look like the following:

```
May I copy sec-server.properties to
$CATALINA_HOME/shared/classes/server y/n? y
--- sec-server.properties moved to
$CATALINA_HOME/shared/classes/server/
```

If you already have `sec-server.properties` in the specified location then you will be prompted to have the application modify it. The modifications will ensure that it points to the correct copy of `sec-server.jks`. The output is as follows:

```
May I update sec-server.properties to point to
$CATALINA_HOME/shared/classes/server/sec-server.jks y/n?
--- Upgraded sec-server.properties
```

The final task performed is the addition of a new property to `oscars.properties`, *aaa.secureCookie*. You must have this property set in `oscars.properties` for the OSCARS web page to work properly and you are not prompted for this change. See section 6.1.2 for more information on this property. The script then completes and you should see the following output:

```
--- Upgrade changes complete
```

You may now run `do_install.sh` to complete installation. The script should do most of the necessary work for you. **When prompted with the following message you should answer ‘n’:**

```
Do you wish to copy key files and oscar's configuration files
to Tomcat now? [y/n] n
```

Your upgraded software is then installed! To complete the upgrade you must read section **7.2 Building Your Inter-Domain Routing Table** for information on using the new inter-domain routing features.

3.3 Installing the TERCE

The TERCE stores XML topology and the static list of local routes used by the IDC. Currently these components are kept in manually generated XML files, but in the future they will be automatically generated. You must install the TERCE to run OSCARS. The TERCE is included in the DCN Software Suite and can only be installed after Axis2 is installed (i.e. after you run `do_build.sh`). The installation can be done by running the following commands:

```
% cd dcn-software-suite-0.3/terce
% ./do_install.sh
```

Configuring the TERCE is detailed in section **5 Describing Your Network Topology**. If this is your first time installing the software then continue to section **3.4 Configuring OSCARS with MySQL**.

3.4 Configuring OSCARS with MySQL

Before you can run your OSCARS installation for the first time you must tell it the username and password it can use to access the database. This should be the same username and password you specified when running `do_build.sh`. You set these properties in the file `$CATALINA_HOME/shared/classes/server/oscars.properties`. The properties are as follows:

| Property | Example Value | Description |
|--------------------------------------|---------------|---|
| hibernate.connection.username | oscars | The MySQL username the OSCARS IDC uses to access the database |
| hibernate.connection.password | mypass | The MySQL password the OSCARS IDC uses to access the database |

(Note: Hibernate is the name of the library used to manage database connections.)

The other properties in this file do not need to be changed to verify OSCARS installed correctly, but you will need to configure them once you want to begin using OSCARS to create circuits. A detailed description of the available properties can be found in section **6.1 Configuring oscar.properties**. If this is your first time installing the software then continue to section **3.5 Creating the First User Account**.

3.5 Creating the First User Account

You need to create a user account so you can login to the OSCARS web page, called the web user interface (WBUI), and verify your installation. You must create the first account with a provided command-line script but additional users can also be created from the WBUI. The tool to create a user, `idc-useradd`, can be found in the `tool/utlis` directory. The commands to create the first user are as follows:

```

% cd dcn-software-suite-0.3/idc/tools/utils
% ./idc-useradd myuser
* indicates a required field
Login*: [oscars@hopishib utils]$ ./idc-useradd
* indicates a required field
Login*: oscars-admin
Password*:
Confirm Password*:
First Name*: John
Last Name*: Smith
Cert Subject:
Cert Issuer:

1. Energy Sciences Network
2. Internet2
Select the user's organization (by number): 2

1. OSCARS-user
2. OSCARS-engineer
3. OSCARS-administrator
4. OSCARS-service
Select the user's role(s) (numbers separated by spaces): 2 3
Personal Description:
Email(Primary)*: myemail@mydomain.net
Email(Secondary):
Phone(Primary)*: 5555555555
Phone(Secondary):

```

The fields with a * are required. It is recommended at this time you just fill-in the minimum requirements. You can always modify this information later when you log-in to the WBUI. It's important to note that the user you create should have at least **OSCARS-administrator** privileges so it can create new users. You may also want to give it **OSCARS-engineer** privileges so you can view and manage all circuits. After creating this user you are ready to test your installation. If this is your first time installing the software then continue to section **3.6 Verifying the Web Service API Installation**.

3.6 Verifying the Web Service API Installation

You may verify that the web service API deployed correctly through the Axis2 administrator interface by doing the following:

1. Open <http://your-machine:8080/axis2/axis2-admin/> in your browser
2. Log- in to the page that loads (default user: **admin**, default password: **axis2**)
3. Click "Available Services" on the left-hand side of the screen

You should see an entry for “OSCARS” on the page that loads, followed by a list of web service calls. This indicates that installation was successful. If you do not see it, try re-installing the IDC.

3.7 Verifying the Web User Interface (WBUI) Installation

The Web User Interface (WBUI) is a set of web pages provided with OSCARS for managing reservations and users. To verify that the WBUI installed correctly visit the following URL:

- <http://your-machine:8080/OSCARS>

A login page should appear. You can login using the account you created in section **3.5 Creating the First User Account**. Once logged-in a blank page should load (reservations will be listed on this page once you begin actively using OSCARS). At this time you cannot create circuits until you do more configurations but you can manage users (see section **4 Creating and Managing User Accounts** for more information).

4 Creating and Managing User Accounts

The OSCARS IDC has a built-in system for authenticating and authorizing requests. User information is kept in the MySQL *aaa* database and web service requests are authenticated using X.509 certificates. Creating user accounts requires issuing certificates and setting-up user accounts via the WBUI.

4.1 Adding Users to the Database

Users and permissions are stored in a MySQL database. This section details how to add users to the database and assign them permissions.

4.1.1 Creating a New User Accounts

You can create new users from the OSCARS web page called the web user interface (WBUI). The five steps for creating a new user via this method are:

1. Visit <http://your-server:8080/OSCARS> – where *your-server* is the name of the server on which the WBUI is running
2. Login with a user account that has administrator privileges (such as the one you created in **3.5 Creating the First User Account**)
3. Click the **Users** tab on the top of the page that loads
4. Click the **Add User** button
5. Complete all the fields outlined in green. Most of the fields are self-explanatory but a few are worth mentioning:

- *X.509 Subject Name* – Use this field to copy an X.509 certificate subject of a user. The subject of a certificate can be determined with `keytool -list -keystore your-keystore.jks -alias yourcertalias -v`.
- *X.509 issuer name* – Generally the X.509 Subject Name is enough but you may also use this field to indicate the issuer of the certificate. In most cases you do not need to use this field.
- *Choose Role(s)* – These determine what permissions users have. More complicated permissions are beyond the scope of this document.

6. Click the **Add** button on the top of the screen

4.1.2 Modifying Users

You may modify user accounts via the WBUI under the **Users** tab. Clicking on the users last name will display a form for editing the user's profile. The form should be self-explanatory as it is similar to the add users form.

4.1.3 Deleting Users

You may delete user accounts via the WBUI under the **Users** tab. Clicking on DELETE will remove the user after a confirmation.

4.2 Managing X.509 Certificates

X.509 certificates are passed in web service messages to authenticate users. **You do not need to create any X.509 certificates to setup circuits on the local domain via the WBUI. You may skip this section until you are ready to begin accepting requests from applications using the web service API or accepting requests from other IDCs.** Each certificate you accept needs to be signed by a certificate authority (CA) that the server trusts. This section describes how to use an external CA and how to become your own CA.

4.2.1 Keystores

The OSCARS IDC keeps certificates in keystore files. These files can be accessed by running the `keytool` command. The important keystores are:

- *sec-server.jks* – stores root certificates used to authenticate user requests. Located in `$CATALINA_HOME/shared/classes/server`.
- *sec-client.jks* – stores certificates used to send requests. Located in `$CATALINA_HOME/shared/classes/repo`.

- *ssl-keystore.jks* – stores CA certificates for authenticating servers contacted that run SSL. Located in \$CATALINA_HOME/shared/classes/repo. **NOTE: The password for this keystore MUST BE ‘oscars’ for current versions of the software.**

4.2.2 Using an External CA

You may have user certificates signed by external CAs. You may see which CA certificates are installed by running the following command:

```
% keytool -list -keystore $CATALINA_HOME/shared/classes
/server/sec-server.jks -V
```

Installing new CA certificates requires you to run `keytool` and rebuild the OSCARS IDC code (this is likely to change in the next release). Run the following command to import a new root certificate:

```
% keytool -import -keystore $CATALINA_HOME/shared/classes/server
/sec-server.jks -alias NewCA -file path-to-cert/ca.cer
```

Don't forget to rebuild the code after this step.

4.2.3 Generating Your Own CA

You may also generate your own CA certificate. This is optional and requires the OpenSSL library. There are various issues with safeguarding root certificates that are beyond the scope of this document. The five basic steps to allow you to generate a root certificate are:

1. Create a directory for your CA certificates

```
% mkdir ca_certs
```

2. Create an openssl.conf file in the new directory (search the Internet for examples)

3. Create files for tracking created certificates in directory created in step 1

```
% touch certindex.txt
% echo '100001' > serial
```

4. Create a private key

```
% openssl genrsa -des3 -out ca_private.key 1024
```

5. Create a public key certificate from the private key

```
% openssl req -new -x509 -days 3650 -key ca_private.key -out ca.cer
```

4.2.4 Signing User ‘Certificate Signing Requests’ (CSRs)

If you do decide to run your own CA certificate, you will be responsible for signing user Certificate Signing Requests (CSRs). This requires the **user** to run the following commands:

```
% keytool -genkey -alias user1 -keystore sec-client.jks -storepass
password -validity 3650
% keytool -certreq -alias user1 -keystore sec-client.jks -file
~/oscars_certs/dcstest_ca/requests/user1.csr
```

Then, the user sends you the CSR file (`user1.csr`). You can then sign the CSR and convert it to an X.509 certificate with the following commands:

```
% openssl ca -config openssl.conf -cert ca.cer -in
requests/user1.csr -keyfile ca_private.key -days 3650 -out
signed_certs/user1.cer
% openssl x509 -in signed_certs/user1.cer -out signed_certs/user1-
X509.cer
```

Afterwards, you can send the user the signed certificate and they can import it **into the same keystore they used to generate the CSR** with the following command:

```
% keytool -import -keystore sec-client.jks -alias user1 -file user1-
X509.cer
```

5 Describing Your Network Topology

5.1 Generating an XML Topology Description

OSCARS currently requires you to manually generate an XML file that describes your network’s topology in the Open Grid Forum (OGF) Network Measurement Working Group (NMWG) control plane topology schema². The topology description you generate describes what is *possible* on your network. For example, it is not concerned with what VLANs are currently provisioned on a network, rather the possible VLANs that could be provisioned on the network.

You can generate both an inter-domain topology description and an intra-domain topology description. The inter-domain topology description will be advertised to other networks, and the intra-domain topology description will be used for internal purposes. Currently, it is recommended you use the same topology for both. Generating the XML topology description is currently one of the most time-consuming portions of the install process.

² <http://anonsvn.internet2.edu/svn/nmwg/trunk/nmwg/schema/>

5.1.1 Example XML files

The easiest way to generate this file is to start from the two examples provided with the DCN Software Suite. Both files describe the same topology of the “blue pod” used in Internet2 DCN workshops³. You can find the example XML files in the following locations on your system:

- `$CATALINA_HOME/shared/classes/terce.conf/tedb-inter.xml`
- `$CATALINA_HOME/shared/classes/terce.conf/tedb-intra.xml`

Both contain the same topology so either will be fine. You may edit them directly or edit a copy in another location on your system.

5.1.2 Domains, Nodes, Ports and Links

The NMWG topology schema consists of a hierarchy of domains, nodes, ports, and links. The table below describes each of these elements.

| Element | Child Element | Description |
|---------------|---------------|--|
| domain | node | Represents an administratively-similar set of devices. |
| node | port | Represents a network device. For this installation, it represents a VLSR |
| port | link | Represent a physical or virtual port on the network. Corresponds to a physical port number and/or DRAGON local-id for this installation. |
| link | - | Represents a connection between two ports. For the purposes of this installation, you will likely have one port per link. |

5.1.3 Fully-Qualified Identifiers

Every element in the domain-node-port-link hierarchy has an “id” attribute. The ID takes the value of a Uniform Resource Name (URN) that contains not only an ID for the element defining it, but also its parent elements. This type of identifier is referred to as a fully-qualified identifier. These IDs always begin with the prefix “urn:ogf:network:”. This prefix is followed by a colon-delimited list of identifiers appropriate for that hierarchical level. For example, a fully-qualified port ID contains a domain ID, a node ID, and the port ID. The hierarchical level of each portion is indicated by either a “domain=”, “node=”, “port=”, or “link=” prefix. Examples of different

³ <http://events.internet2.edu/2007/DCN/>

fully-qualified link ID types from the example topology files that come with the software are shown below:

| Type | Fully-Qualified Identifier |
|------------------|---|
| domain ID | urn:ogf:network:domain=blue.pod.lan |
| node ID | urn:ogf:network:domain=blue.pod.lan:node=vlsr1 |
| port ID | urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port=3 |
| link ID | urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port=3:link=11.2.1.2 |

5.1.4 <topology> Element

The <topology> element is the top-level element of the XML description. It contains the following important attributes and values.

| Element/Attribute | Example | Description |
|-------------------|---|--|
| <i>Id</i> | blue-topology | An identifier for this element. It has no significance currently and may be any string. |
| <idcId> | https://idc.blue.pod.lan:8443/axis2/services/OSCARS | The URL to the IDC advertising this topology. In most cases, you will only need to replace 'idc.blue.pod.lan' with the host on which you have installed the IDC. |

5.1.5 <domain> Element

The <domain> element contains a set of commonly administered nodes. In addition to < node >, <domain> contains the following attributes and children elements:

| Element/Attribute | Example | Description |
|-------------------|-------------------------------------|---|
| <i>Id</i> | urn:ogf:network:domain=blue.pod.lan | A URN representing the domain's ID. The portion after "domain=" MUST be globally unique and SHOULD take the form of a DNS name. |

5.1.6 <node> element

The <node> element represents a VLSR. In addition to a list of < port > elements, <domain> contains the following attributes and children elements:

| Element/Attribute | Example | Description |
|-------------------|--|---|
| <i>Id</i> | urn:ogf.network:domain=blue.pod.lan:node=vlsr1 | A URN representing the node's ID. The portion after "node=" may be any valid string and is not required by the IDC to have any specific value. |
| <address> | 192.168.2.4 | An IP address that corresponds to the "router-id" field in the VLSR's ospfd.conf file. See DRAGON install documents for more info about router-id's and ospfd.conf. |

5.1.7 <port> Element

The <port> element represents a connection point between VLSRs. **The ID of this element must be set a specific way for OSCARS to correctly provision circuits.** The ID field corresponds to a physical port and/or local-ID controlled by DRAGON. The ID attribute's format is different depending on the network devices being controlled.

| Element/Attribute | Example | Description |
|-----------------------------|---|--|
| <i>id</i> | urn:ogf.network:domain=blue.pod.lan:node=vlsr1:port=3 | This type of ID maps directly to the local-ID or physical port of an Ethernet switch. |
| <i>Id</i> | urn:ogf.network:domain=anna.internet2.edu:node=vlsr1:port=1-2-1 | This type of ID maps to a port on an Ethernet switch. The format is (chassis+shelf)-slot-subport |
| <i>Id</i> | urn:ogf.network:domain=dcn.internet2.edu:node=CHIC:port=S23647 | This type of ID map is applied to ports on an ESLM card in a Ciena CoreDirector. The format is S(Subnet Interface ID). See DRAGON documentation for more information about subnet interface IDs. |
| <i>Id</i> | urn:ogf.network:domain=dcn.internet2.edu:node=CHIC:port=DTL1 | This type of IDC maps to internal SONET links between CienaCoreDirectors. The DTL values map to those in the DRAGON configuration files. |
| <capacity> | 1000000000 | Maximum capacity of the port in bits per second |
| <maximumReservableCapacity> | 1000000000 | Maximum amount of capacity that can be reserved on a link in bits per second. It MUST be <= capacity |

| | | |
|--|-----------|---|
| <minimumReservableCapacity> | 100000000 | Minimum amount of capacity that can be reserved on a link in bits per second. It MUST be <= maximumReservableCapacity |
| <granularity> | 100000000 | The minimum increment in which bandwidth can be reserved. |

5.1.8 <link> Element

| Element/Attribute | Example | Description |
|--|---|---|
| <i>id</i> | urn:ogf.network:domain=blue.pod.lan:node=vlsr1:port=3:link=11.2.1.2 | The link portion of this ID should map to the TE address used in the DRAGON ospfd.conf files. For Ethernet interfaces on the edge of your network the link-id is only used as a label and may be any value you wish (i.e. 1, *, myport, etc) |
| <remoteLinkId> | urn:ogf.network:domain=*.node=*.port=*.link=* | The remote link to which the link is connected. Values may all be "*" if connected to an end-host or domain without a topology description. |
| <trafficEngineeringMetric> | 1000000000 | A metric associated with this link |
| <capacity> | 1000000000 | Maximum capacity of the link in bits per second. Must be <= the parent <port> capacity |
| <maximumReservableCapacity> | 1000000000 | Maximum amount of capacity that can be reserved on a link in bits per second. It MUST be <= capacity |
| <minimumReservableCapacity> | 100000000 | Minimum amount of capacity that can be reserved on a link in bits per second. It MUST be <= maximumReservableCapacity |
| <granularity> | 100000000 | The minimum increment in which bandwidth can be reserved. |

5.1.9 <switchingCapabilityDescriptors> Element

This element is a child of link and contains information about its parent's switching capability. This information can be derived from DRAGON's ospfd.conf or narb.conf.

| Element/Attribute | Example | Description |
|--------------------|----------|---|
| <switchingcapType> | l2sc | “l2sc” for Ethernet links and “tdm” for SDH/SONET links |
| <encodingType> | ethernet | “Ethernet” for Ethernet links and “sdh” for SDH/SONET links |

5.1.10 <switchingCapabilitySpecficInfo> Element

This element is a child of <switchingCapabilityDescriptors> and contains information specific to the switching capability type. Currently, only elements for l2sc links are defined.

| Element/Attribute | Example | Description |
|---------------------------|-------------------------|---|
| < interfaceMTU > | 9000 | The maximum transmission unit (MTU) of this link |
| < vlanRangeAvailability > | 0, 3000-3100, 3150-3200 | The range of VLANs available for provisioning on a link. Use a “-” to separate continuous ranges and a “,” to separate discontinuous ranges. A value of 0 indicates that untagged interfaces are allowed on the interface represented by this link. Untagged interfaces will not be allowed if a value of 0 is not in the range. |

5.2 Creating a Static List of Local Paths

The current version of the OSCARS IDC requires you to statically generate intra-domain paths in an XML file. You only need to define those paths that are between the edges of you network. Defining inter-domain paths that extend beyond the local network are discussed section **7 Inter-Domain Configuration**. An example of the static paths file can be found at the following location on your system:

- \$CATALINA_HOME/shared/classes/terce.conf/static-routes.xml

You may edit this file directly or make a copy. Here are some helpful guidelines when populating this file:

- < staticPathEntry> elements contain the paths. You will need a <staticPathEntry> for every source/destination combination you wish to use (including separate paths for the forward and reverse direction).
- < srcEndpoint> and < destEndpoint> elements contain **fully-qualified link-IDs**. Every time these link-IDs are seen, the associated path will be returned. If a lookup service name is used, it will get converted to a fully-qualified link-ID before making the path calculation.

- The <path> element carries the path to be returned when the associated source and destination are given by the user or as the ingress and egress to your network. The <path> contains a list of <hop> elements. Each of these <hop> elements contains a <linkIdRef>, which is a fully-qualified link ID.
- Each local hop in a path should be followed by the link to which it is connected. Every local hop should be indicated in the path.

NOTE: Previous versions of OSCARS required both intra-domain and inter-domain paths to be specified in static-routes.xml. You can no longer define inter-domain paths in static-routes.xml, see section 7.2 Building Your Inter-Domain Routing Table for information on defining inter-domain routes.

5.3 Configuring the `terce-ws.properties` file

NOTE: You may skip this section if your `static-routes.xml`, `tedb-intra.xml`, and `tedb-inter.xml` file are in `$CATALINA_HOME/shared/classes/terce.conf`.

The TERCE is a web service that acts as the topology exchange and route computation element for OSCARS. In the future, it will act as the intermediary between OSCARS and the NARB, but only static topology description and route files are supported in this release. To ensure the TERCE properties file can locate the static topology and route file you have generated, it must be edited as follows:

1. Open `$CATALINA_HOME/shared/classes/terce.conf/terce-ws.properties` in a text editor
2. Change the properties file to look like the following (where *location-of-your-topology-file* is replaced with the custom path to your topology file and likewise for *location-of-your-static-routes-file*)

```
#static tedb properties

tedb.type=static

tedb.static.db.interdomain=location-of-your-topology-file
tedb.static.db.intradomain=location-of-your-topology-file


#static rce properties

rce.type=static

rce.static.file=location-of-your-static-routes-file
```

5.4 Populating the Scheduling Database

The final step is to import the topology information from your XML file into the OSCARS scheduling database so that it can keep track of which resources are used on the network. This is done by defining your local domain in the database and running `updateTopology.sh`.

5.4.1 Defining Your Local Domain

You must define your local domain so that OSCARS know which links are on your network. This information is stored in MySQL under the `bss.domains` table but you add it using the `idc-domainadd` script. You may run this script with the following commands:

```
% cd dcn-software-suite-0.3/idc/tools/utils/
% ./idc-domainadd
Topology Identifier (i.e. mydomain.net)*: blue.pod.lan
IDC URL*: https://your-machine:8443/axis2/axis2-admin/
Descriptive Name (for display purposes)*: Blue Pod
Abbreviated Name (for display purposes)*: blue
Is this your IDC's local domain? [y/n] y
```

The most important part to note is that the first field, Topology Identifier, match the domain ID you use in your URNs.

5.4.2 Run `updateTopology.sh`

The `updateTopology.sh` script syncs the database with your Intra-Domain topology file. The three steps for running this script are:

1. Verify Tomcat is running
2. Change your current directory to `dcn-software-suite-0.3/idc/tools/updatedbterce`

```
% cd dcn-software-suite-0.3/idc/tools/updatedbterce
```
3. Finally, run the `updateTopology.sh` script and point it to your TERCE service:

```
% ./updateTopology.sh http://127.0.0.1:8080/axis2/
services/TERCE
```

If no errors are returned, your database is now populated with the appropriate topology information.

6 Preparing for Circuit Creation

After creating user accounts and describing your topology, there are a few final configuration steps required before you can begin creating circuits. This section describes configuration of the `oscars.properties` file, inter-domain settings, and running the automated scheduler.

6.1 Configuring `oscars.properties`

The `oscars.properties` file is the main area in which the OSCARS IDC retrieves installation-specific settings. These include settings for accessing the MySQL database, AAA, the perfSONAR Lookup Service, interacting with DRAGON, and more. The `oscars.properties` file is located on your system at:

- `$CATALINA_HOME/shared/classes/server/oscars.properties`

Your installation comes with a default `oscars.properties` file. This subsection details the properties required for an installation of the OSCARS IDC on a DRAGON-controlled network.

6.1.1 MySQL Database Properties

The required properties related to the MySQL database are:

| Property | Example Value | Description |
|--------------------------------------|---------------|---|
| hibernate.connection.username | oscars | The MySQL username the OSCARS IDC uses to access the database |
| hibernate.connection.password | mypass | The MySQL password the OSCARS IDC uses to access the database |

(Note: Hibernate is the name of the library used to manage database connections.)

6.1.2 Authentication, Authorization, and Accounting (AAA) Properties

The required properties related to AAA are:

| Property | Example Value | Description |
|---------------------|---------------|---|
| aaa.salt | os | The value with which encrypted passwords are salted. A value of 'os' means a password could be generated with <code>crypt('password','os')</code> |
| aaa.userName | oscars | The username for accessing a secure cookie. Required if running the WBUI. It may be any valid string value. |

| | | |
|----------------------------|------------|---|
| aaa.sessionName | oscarssess | The session name for a secure cookie. Required if running the WBUI. It may be any valid string value. |
| aaa.secureCookie | 1 | If set to 1 it will require users to use SSL to access the WBUI. For ease of installation this defaults to 0 but it is recommended you change this value to 1 after setting-up SSL. |
| aaa.useSignalTokens | 1 | If set to 1 then signaling tokens will be generated for each createReservation request. Signaling tokens can be transferred between a user who sends a createReservation request and another who wants to send the createPath. Note: If you running the IDC on a PowerPC you MUST set this property to 0. |
| aaa.guestLogin | guest | The value of this optional property is a username that can be used to assign a guest login to the WBUI. The difference between a guest and a regular user is that a guest can login to the WBUI from multiple machines. In the example given the user “guest” has this ability. An IDC administrator should be very careful about what permissions it gives a guest account as it is NOT secure. |

6.1.3 Topology Exchange and Pathfinding Properties

The required properties related to topology exchange and pathfinding are:

| Property | Example Value | Description |
|------------------------------|---------------|--|
| pathfinder.findPath | 1 | In general, always set to 1. If set to 0, the user must always provide the path. |
| pathfinder.pathMethod | terce | Always set to ‘terce’ for DRAGON installations. This property indicates the pathfinding component to use. |
| tedb.tedbMethod | terce | Always set to ‘terce’ for DRAGON installations. This property indicates the traffic engineering database type (i.e. where to get topology info). |

| | | |
|------------------|--|--|
| terce.url | http://127.0.0.1:8080/axis2/services/TERCE | The URL to the TERCE service. In most cases, the example URL can be used exactly as shown. |
|------------------|--|--|

6.1.4 Path Setup Properties

The required properties related to path setup are:

| Property | Example Value | Description |
|-----------------------------------|--------------------------|--|
| pss.method | dragon | The method for setting up circuits. A value of 'dragon' means that the IDC will try to use DRAGON to create the path. A value of 'stub' simulates circuit setup but does not perform any action. |
| pss.dragon.password | dragon | The telnet password used to access the dragon VLSR. |
| pss.dragon.ssh.portForward | 1 | Sends telnet traffic over an SSH tunnel if set to 1. If 0, telnet traffic is sent directly to VLSR. |
| pss.dragon.setERO | 0 | Explicitly sets every link on the local path if set to 1. This is highly recommended as it ensures DRAGON uses exactly the same path reserved in OSCARS. You can only set this value to 1 if you are running the version of DRAGON in DCNSS 0.3 or later. |
| pss.dragon.ssh.user | oscars | Required if pss.dragon.ssh.portForward= 1. The SSH user must be the same for all VLSR machines. |
| pss.dragon.ssh.key | /home/oscars/.ssh/id_rsa | Required if pss.dragon.ssh.portForward= 1. The public key used for SSH login must be the same for all VLSR machines. |

| | | |
|--------------------------------------|-------------|---|
| | | See ssh-keygen man pages for more information. |
| pss.dragon.remotePort | 2611 | The telnet port on which the DRAGON CLI is running. Most DRAGON installations will use 2611. |
| pss.dragon.nodeId⁴ | 127.0.0.1 | The address used by telnet to access the VLSR with <i>nodeId</i> . If pss.dragon.ssh.portForward= 1, you should set this to 127.0.0.1. |
| pss.dragon.nodeId.ssh | 192.168.2.4 | Required if pss.dragon.ssh.portForward= 1. The SSH address to access a VLSR with the given <i>nodeId</i> . |
| pss.dragon.promptPattern | vlsr | Optional. Defaults to “vlsr”. This value is a string that will be used in the telnet prompt of all the VLSRs. You will get a large hexdump in your logs if this property is not set properly. |

6.1.5 Other Properties

Other required properties are:

| Property | Example Value | Description |
|--------------------------|--|--|
| logging.rsvlogdir | /usr/local/tomcat/logs | Location to keep log information on individual reservations. |
| lookup.url | http://packrat.internet2.edu:8009/perfSONAR_PS/services/LS | A URL to a perfSONAR Lookup Service |
| mail.webmaster | webmaster@blue.pod.lan | Email address of OSCARS administrator who will receive |

⁴ Not required if pss.dragon.ssh.portForward= 0 AND the nodeId field in the XML topology description is a routable address that can be used by telnet.

| | | notifications from the server |
|------------------------|---------------------------|---|
| mail.recipients | user1@my.net:user2@my.net | A colon delimited list of recipients of email notifications from OSCARS |

6.2 Running the Scheduler

The scheduler is a script that automatically builds circuits for users that do not wish to use signaling and deletes expired reservation. Running the scheduler is required for the IDC to function properly. You may run the scheduler with the following command:

```
% cd dcn-software-suite-0.3/idc/tools/scheduler
% nohup ./scheduler.sh > ~/scheduler.log &
```

The above command will run the scheduler in the background and log DRAGON output in scheduler.log. This assumes you chose to build the OSCARS tools when you ran do_build.sh. If you do not choose to compile the tools you may do so by simply running the command ant in the scheduler directory.

6.3 Using Hostnames Instead of URNs in Requests

An optional step that will make provisioning easier is to generate hostnames for the URNs on the edge of your network. This will prevent users from having to enter long URNs into requests for circuits on your network. Currently this functionality is provided by a **Lookup Service**. The Lookup Service is analogous to DNS in the IP world but is centralized at the moment. You may email deploy-dcn@internet2.edu with the list of names and URNs you would like added to the Lookup Service maintained by Internet2. In the future the Lookup Service will have a distributed architecture and each network will run their own copy.

6.4 Verifying that You Can Build Local Circuits

If you have completed all the previous steps in this document you may now try provisioning a local circuit via the WBUI. The following steps will allow you to do this:

1. Open a browser and go to <https://your-server:8443/OSCARS> (or port 8080 if SSL is not installed)
2. Login using a user account you created
3. Click the “Create Reservation” tab
4. Enter the following value into the form (at a minimum):
 - a. Source – a fully qualified link-id or lookup service name of one edge of your network

- b. Destination - a fully qualified link-id or lookup service name of another edge of your network
 - c. Bandwidth – the amount of bandwidth to reserve
 - d. Description – a brief description of your reservation
 - e. VLAN – the VLAN number to use. If you don’t know (or care) which VLAN to use please specify “any.” **NOTE: You must enter a value in the field. The WBUI was originally built to request MPLS tunnels that used IP addresses. Currently the WBUI distinguishes MPLS requests from other requests with this field.**
5. Click the “Reserve Bandwidth” button.

Assuming everything works correctly, you should see a page indicating the reservation was made. Click the “Refresh” button until you see the status of “ACTIVE” indicating that the scheduler built the circuit. Once you see ACTIVE you may begin using your circuit.

7 Inter-Domain Configuration

This section details the steps required to configure your IDC to communicate with other IDCs. The basic process involves adding neighboring domains to the OSCARS database, building your inter-domain routing tables, and generating an X.509 certificate for communication with other IDCs. Once completing this section you will be able to request circuits that span across multiple domains.

7.1 Making your IDC Aware of Other Domains

You must add an entry for each of your neighboring domains to the `bss.domains` table in MySQL. This is can be done using the `idc-domainadd` script and the following commands:

```
% cd dcn-software-suite-0.3/idc/tools/utils/
% ./idc-domainadd
Topology Identifier (i.e. mydomain.net)*: red.pod.1an
IDC URL*: https://their-machine:8443/axis2/services/OSCARS
Descriptive Name (for display purposes)*: Red Pod
Abbreviated Name (for display purposes)*: red
Is this your IDC's local domain? [y/n] n
```

Replace ‘red.pod.1an’ with the local part of your neighbor’s domain-id, ‘red’ with a string for your neighbor (can be anything), and add the URL of their IDC. You will also need to create a user account for each neighboring domain using the process described in section 4 *Creating and Managing User Accounts*. The most important piece is that the `certSubject` field matches the X.509 certificate subject they have provided.

7.2 Building Your Inter-Domain Routing Table

Currently OSCARS maintains static routing tables that determine the path a circuit will take to reach another domain. You can build these routing tables using the `idc-route` command found in the “tools/utls” directory. The command allows you to indicate a destination in another domain and the “path” you will use to reach that destination. A destination may be a domain, node, port or link URN. The path to your destination may be as simple as the egress link from your domain (a *loose* inter-domain path) or it may be as specific as every ingress and egress in the path to the destination (a *strict* interdomain-path). The types of routes you enter depend on the needs of your network. Every IDC administrator should read **Section 7.2.1** Populating the `<remoteLinkId>` fields in `tedb-intra.xml` and `tedb-inter.xml` then use the list below to help decide which other sections to read and what types of routes to add:

1. If your IDC is going to manage a regional network, campus, or lab with a **single** connection to another domain running the DCN Software Suite 0.3+ (i.e. Internet2 DCN) see **7.2.2 Defining a Default Route**.
2. If your IDC is managing a network with **multiple** connections to other domains running the DCN Software Suite 0.3+ see **7.2.3 Defining a .**
3. If your IDC is managing a network with one or more connections to domains running a version of the DCN Software Suite older than 0.2 or are running their own version of an IDC see **7.2.4 Defining a .**

NOTE: In previous versions of OSCARS static-routes.xml maintained inter-domain paths as well as intra-domain path. It still maintains intra-domain paths but it does not maintain inter-domain paths. This means that you will need to add routes as indicated in this section when upgrading to 0.3 from a previous version. This new method does NOT require you to specify the exact path to every domain so this will hopefully be much easier than building your static-routes.xml table.

7.2.1 Populating the `<remoteLinkId>` fields in `tedb-intra.xml` and `tedb-inter.xml`

The first step to adding inter-domain routes is to update your topology description with your neighboring domains’ *remoteLinkIds*. You do not need to add them all at once, you can add them as you get them and the steps will be the same. First, to obtain the *remoteLinkIds* you must contact the IDC administrator of your neighboring domain (via phone, email, etc) and exchange this information. You will need to give them the fully-qualified link-id that represents the interface on your network that connects to their network. They will do the same for you and you will add the information they give you to the *remoteLinkId* field in `tedb-intra.xml` and `tedb-inter.xml` of the link you gave them. You will then run `updateTopology.sh` to update your database with this new information. The commands for this are as follows:

```
% cd dcn-software-suite-0.2/idc/tools/updatedbterce
```

```
% ./updateTopology.sh
http://127.0.0.1:8080/axis2/services/terce
```

7.2.2 Defining a Default Route

A default route is an inter-domain route that will be taken when a destination does not match any other entries in the routing tables. If the default route is the only route in the table then it will ALWAYS be used. This may be desirable when your IDC manages a network that only has a connection to one other network. Below is an example of how you may add a default route to the OSCARS routing tables:

```
% cd dcn-software-suite-0.3/idc/tools/utils
% ./idc-route add -default -egress
urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port=6:link=11.
1.11.2
```

The above command states that if someone requests a circuit in another domain it will exit the network on *urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port=6:link=11.1.11.2* if no other entry matches. If this is the only entry made to the routing tables ALL requests with a destination outside the domain will exit using the specified egress. Assuming this is the only entry, by looking at the example topology provided with the TERCE we see that this means requests will always be forwarded to the “red.pod.lan” domain. This can be seen by looking at the *remoteLinkId* field in the definition of the link is tedb-intra.xml. Issue the above command and replace the value given to *-egress* with a link-id on your network to create a default route.

7.2.3 Defining a Path with Only an Egress

You should define this type of path if you have multiple connections to other networks capable of accepting loose inter-domain paths (LIDP). A LIDP is one that does not contain the ingress and egress link of every domain in a path. You may define an LIDP containing only the egress of your network to use. The path passed to the next domain will include this egress but it will be the responsibility of the next domain to fill in any additional inter-domain hops between itself and the destination. All domains running version 0.3 or greater of the DCN Software Suite have this capability. Below is an example of a command used to create this type of route:

```
% cd dcn-software-suite-0.3/idc/tools/utils
% ./idc-route add -dest urn:ogf:network:domain=red.pod.lan -
egress
urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port=6:link=11.
1.11.2
```

The above command states that any destination with a domain id of *red.pod.lan* will be routed to the next domain on the given egress link. Notice that the destination is a domain-id as opposed to a link-id. This means that any link-id that contains “domain= red.pod.lan ” will match this entry. For example, *urn:ogf:network:domain=red.pod.lan:node=vlsr1:port=5:link=11.1.11.3* would

match this entry if given as a destination. The destination field in the routing table may be a domain-id (as shown), node-id, port-id, or link-id. Destinations in requests will match the route for which there is the most specific definition. In other words, if one route matches at the node level and another matches at the domain level, then the node level match will be used. You may want to match at levels more specific than domain-d if you have multiple ways to get to a domain and it makes more geographic sense to take a particular egress when going to one node, port, or link than others. You may also make the routing decision based on the source in your domain. **Run `./idc-route -help` for more information.**

7.2.4 Defining a Path with Multiple Hops

You may define a route that indicates more than one inter-domain hop used to get to a destination. You should use this type of entry when talking to an IDC that requires a strict inter-domain path (SIDP). A SIDP is a path with the ingress and egress hop for each domain. In some cases you may also define a loose inter-domain path with multiple hops, but that does not necessarily contain every hop in the path. Below is an example of a command used to define an SIDP:

```
% cd dcn-software-suite-0.3/idc/tools/utils
% ./idc-route add -dest urn:ogf:network:domain=green.pod.lan -
multi
Enter local egress:
urn:ogf:network:domain=blue.pod.lan:node=vlsr1:port=6:link=11.
1.11.2
Enter hop URN or type 'end' to complete:
urn:ogf:network:domain=red.pod.lan:node=vlsr3:port=6:link=11.1
.11.1
Enter hop URN or type 'end' to complete:
urn:ogf:network:domain=red.pod.lan:node=vlsr1:port=5:link=11.1
.11.3
Enter hop URN or type 'end' to complete:
urn:ogf:network:domain=green.pod.lan:node=vlsr1:port=5:link=11
.1.11.4
Enter hop URN or type 'end' to complete: end
Route Added
```

In the above example a SIDP is defined to *green.pod.lan*. The first element specified is the egress from the local domain (*blue.pod.lan*). The next link is the ingress to an intermediate domain (*red.pod.lan*). This link **MUST** be the same as that defined in the *remoteLinkId* field of your topology description (in *tedb-inta.xml*). The third link defined is the egress of *red.pod.lan* to *green.pod.lan*. The final hop is the ingress to *green.pod.lan*. Notice you do not need to define the egress in *green.pod.lan*. This is because the egress will be the destination in the user's request.

This is the basic way in which you define a path with multiple hops. There are more advanced uses of this functionality, including LIDPs that include domain, node, or port identifiers (as opposed to link identifiers) but this is saved for a more advanced document on inter-domain routing in OSCARS.

7.3 Generating a Server Certificate for Inter-Domain Requests

You must generate a certificate that your domain will pass to other domains. This certificate is stored in the following keystore file:

- `$CATALINA_HOME/shared/classes/repo/sec-client.jks.`

These steps will create a certificate for your domain:

1. Generate a certificate and certificate signing request (CSR):

```
% cd $CATALINA_HOME/shared/classes/repo/
% keytool -genkey -alias myidc -keystore sec-client.jks -storepass
password -validity 3650
% keytool -certreq -alias myidc -keystore sec-client.jks -file
~/myidc.csr
```

2. Send the CSR to a CA for signing. If you are running your own CA, see section **4.2.4 Signing User ‘Certificate Signing Requests’ (CSRs)**.

3. Install the signed certificate into your keystore:

```
% cd $CATALINA_HOME/shared/classes/repo/
% keytool -import -keystore sec-client.jks -alias myidc -file myidc-
signed.cer
```

4. Modify `$CATALINA_HOME/shared/classes/repo/axis2.xml` to reference your certificate by changing the `<user>` element to the alias of your certificate in the keystore:

```
...
<parameter name="OutflowSecurity">
  <action>
    <items>Timestamp Signature</items>
    <user>myidc</user>
  </action>
</parameter>
...
```

5. Send the subject of your certificate to your neighboring domains. They will use this subject to create a user account for you as described in section *4 Creating and Managing User Accounts*. Also, verify they have the CA certificate of the entity that signed your certificate installed. You may view the subject of your certificate with the following command:

```
% keytool -list -keystore sec-client.jks -alias myidc -V
```


7.4 SSL Certificates

If a neighboring domain runs SSL, you will need to have at least the root of their SSL certificate installed. New SSL certificates can be installed using `keytool` in the following location:

- `$CATALINA_HOME/shared/classes/repo/ssl-keystore.jks`

The commands to import a new SSL certificate are:

```
% cd $CATALINA_HOME/shared/classes/repo/
% keytool -import -keystore ssl-keystore.jks -alias redssl -file
redssl.cer
```

The default password of this keystore is ‘oscars’. **NOTE: The password of this keystore MUST BE ‘oscars’ for current versions of the software.**

8 Advanced Installation

8.1 Axis2

Axis2 provides the IDC with a web service framework. It is installed in Tomcat as its own web application. This section details installation and configuration of Axis2.

8.1.1 Download and Installation

Download Axis2 from the project’s web site at:

- http://ws.apache.org/axis2/download/1_3/download.cgi

You only need to download the **WAR (Web Archive) Distribution** of the software. **You MUST download version 1.3 for the IDC to function properly.**

Unpack and install the downloaded components with the following commands:

```
% unzip -d axis2-1 axis2-1.3-war.zip
% mv axis2-1/axis2.war $CATALINA_HOME/webapps
```

After moving the WAR file to the correct location, restart the Tomcat server:

```
% $CATALINA_HOME/bin/shutdown.sh
% $CATALINA_HOME/bin/startup.sh
```

8.1.2 Setting the AXIS2_HOME Environment Variable

Once Axis2 is installed, you need to set the `AXIS2_HOME` environment variable with its location. To set this environment variable, issue these commands:

```
% AXIS2_HOME=$CATALINA_HOME/webapps/axis2/WEB-INF
% export AXIS2_HOME
```

You may permanently set this variable (recommended) by adding the above commands to the profile file in your home directory (i.e. `.bash_profile` or `.profile`).

8.1.3 Verifying a Successful Installation

Validate that installation was successful by pointing your web browser to the following URL:

- <http://your-machine-name:8080/axis2> **OR**
- <https://your-machine-name:8443/axis2> (if SSL is configured – see section 2.3.5 **Configuring SSL**)

On the page that loads, click **Validate**. A page will appear that indicates the status of your Axis2 installation. If you see a message that reads “The core axis2 libraries are present.” Your installation was successful.

8.2 Rampart

Rampart is an Axis2 module that provides a number of the IDC’s security features. This section details its installation.

8.2.1 Download and Installation

Download the Rampart module from the project’s web site at:

- http://www.apache.org/dyn/mirrors/mirrors.cgi/ws/rampart/1_3/rampart-1.3.zip

Unpack and install rampart with the following commands:

```
% unzip rampart.zip
% cp rampart-1.3/rampart-1.3.mar $AXIS2_HOME/modules/
% cp rampart-1.3/lib/*.jar $AXIS2_HOME/lib/
```

After installing Rampart, you must restart the Tomcat server with the following commands:

```
% $CATALINA_HOME/bin/shutdown.sh
% $CATALINA_HOME/bin/startup.sh
```

8.2.2 Verifying a Successful Installation

You may verify that Rampart was installed correctly through the Axis2 administrator interface. To use this interface point a web browser to the following URL:

- <http://your-machine-name:8080/axis2/axis2-admin/>

You may login using the default username and password (**user: admin, password: axis2**). Click on the **Available Modules** link on the left side of the screen after logging-in. If installation was successful, you will see Rampart in the list that loads.