

***Tivoli Application Dependency
Discovery Manager (TADDM)***
Installation, Customization and Use

Duane Beyer – dbeyer@us.ibm.com

Table of Contents

- Introduction2
- Installing TADDM2
 - Planning for TADDM installation2
 - Starting the TADDM installation3
 - Installing the latest fixes for TADDM3
 - Testing the server4
 - Using the Web interface4
- Customizing TADDM.....5
 - The TADDM Domain Manager5
 - Creating a user.....7
 - The TADDM Product Console.....7
 - Debugging problems starting the Product Console8
 - The collation.properties file8
- Level-1 Discovery.....9
 - Preparing to run a level-1 Discovery9
 - Creating a scope10
 - Running a level-1 discovery10
 - Not what we expected.....12
 - Viewing the results of the Level-1 Discovery.....13
 - Viewing the results with Business Intelligence Reporting Tools (BIRT)13
- Level-2 Discovery.....13
 - Preparing to run a level-2 discovery13
 - Service Account requirements13
 - TADDM access lists14
 - Creating an access list for Linux system discovery14
 - Running a Level-2 Discovery15
 - Known Problems:16
 - Viewing results of the Level-2 Discovery.....16
 - Viewing the topology16
 - Not working as expected.....19
 - Collecting additional data with Custom Servers20
- Level-3 Discovery.....23
 - Preparing to run a level-3 discovery23
- Business applications and business services25
 - Business Applications25
 - Business Services.....25
 - Tracking software levels with TADDM.....26
 - Component Comparison27
- Summary: Experiences with TADDM in our environment29
- Appendix A: TADDM_L1_DISCOVERY BIRT Report31
- Appendix B: Collecting data from z/VM systems.....35
 - zvmprobe setup example35
 - Creating the z/VM Linux guest, named LITSPY35
 - Starting a unique process for TADDM to discover35
 - Creating the Custom Server named zvmprobe36
 - Create a directive file38

Introduction

Tivoli® Application Dependency Discovery Manager (TADDM) discovers services running in your environment and tracks their dependencies. Unlike a number of other tools that require you to install agents on known systems, TADDM uses sensors to probe your environment and discover what is running.

TADDM is marketed as a configuration management tool to help IT administrators monitor application availability, but it is much more than that. TADDM can capture configuration files, track changes and be used as an investigative tool to determine when something has changed in the environment. Another TADDM strength is locating decommissioned servers that were never removed from the environment.

In this paper we discuss our experience installing, configuring and using TADDM in the Linux® Virtual Server (LVS) IT environment.

Installing TADDM

Planning for TADDM installation

Although the *Tivoli Application Dependency Discovery Manager Planning and Installation Guide* is complete, we found the layout to be a bit confusing because it covers a number of different installation scenarios. So it is sometimes hard to identify the steps that you need to take for your particular installation.

In our environment, we installed a TADDM domain manager with remote DB2®. The following list shows the order of activities and documentation topics we followed to set up our environment.

1. Review the entire TADDM planning and installation guide.
2. Set up DB2.
3. Complete the planning worksheet in chapter 2.
4. Starting the server
5. Install TADDM, as described in chapter 5, “Advanced Installation with a Remote DB2 Database.”
6. Test the installation
7. Using the Web interface
8. Using the Product Console
9. Customizing TADDM
 1. Setting up Scopes
 2. Custom Servers

Tivoli Application Dependency Discovery Manager (TADDM)

While the installation of TADDM was fairly straight forward, we recommend reading the *TADDM Planning and Installation Guide* and completing the planning worksheets prior to starting the installation. This involves making decisions about the database to use, as well as the type of TADDM server. TADDM servers could be a leaf node component of an Enterprise Configuration Management Database (CMDB), or a stand-alone TADDM domain server.

In our environment, we installed a stand-alone TADDM domain server with a remote DB2 server. When configured with a remote DB2 server, the instance and archive user must be defined on the DB2 server prior to starting the TADDM Installation. This is detailed in chapter 5 of the *TADDM Planning and Installation Guide* in the section, “Advanced installation with a remote DB2 database,” Steps 1 through 4. When completed, make sure it matches the configuration you entered on the planning worksheet in chapter 2 for the DB2 instance and archive user.

Starting the TADDM installation

As we stated, installation of TADDM is a straightforward task, and is well documented in chapter 5 of the *TADDM Planning and Installation Guide* in the section, “Advanced installation with a remote DB2 database,” starting with Step 5. The GUI installer is started with the shell script `setupLinuxx390.bin`.

We decided not to connect to an IBM Service Management server, (Step 23) nor did we install the WebSphere® Federation Server (Step 25).

If you encounter any problems with the installation, review the log files, which are located in `install_dir/installLogs/`. If you used the default installation location, this will be path `/opt/IBM/cmdb/installLogs/`.

Installing the latest fixes for TADDM

At the time this paper was written, TADDM 7.2 had five available Fix-Packs. It is recommended that you contact your TADDM service team or IBM service representative and request the latest available FixPak. Service was provided to us in the form of a tar file. We simply un-tared the file and ran the `installFixPack.sh` shell script. Installing the latest fixpack resolved a number of problems that we encountered with the base release, including the database dropping, login hangs and Custom Servers not appearing in the server list.

Tivoli Application Dependency Discovery Manager (TADDM)

Testing the server

The default installation directory for TADDM on Linux for IBM System z[®] is /opt/IBM/cmdb. TADDM is started, stopped and status checked with the “control” command located in the /opt/IBM/cmdb/dist/bin directory.

To start the server issue `./control start` from /opt/IBM/cmdb/dist/bin. To check the status of the server, use the “`./control status`” command in the same manner. The control command must be run from the `taddm` user, not root. When all the components of the TADDM server have started, the status will be displayed as follows:

```
taddm:/opt/IBM/cmdb/dist/bin> ./control status
Discover: Started
GigaSpaces: Started
DbInit: Started
Tomcat: Started
Topology: Started
DiscoverAdmin: Started
Proxy: Started
EventsCore: Started
TADDM: Running
```

Additional details about testing the server are covered in chapter 5 of the *TADDM Planning and Installation Guide* in the section, “Testing the Server.”

Using the Web interface

The next step in the *TADDM Planning and Installation Guide* is “Accessing the Product Console.” However, before you can do that you need to start the Web interface. The Product Console Java application is started within a Web interface rather than a command line.

The Web interface runs on a number of supported Web browsers for Linux, UNIX[®], and Windows operating systems. The supported platforms are listed in chapter 2 of the *TADDM Planning and Installation Guide* in the section, “Supported browsers for the Product Console and Domain Manager.”

From a supported Web browser, navigate to **`http://taddm_server_ip:9430`**, where `taddm_server_ip` is the IP address or the fully qualified host name for the TADDM server. This will access the launch page for TADDM. The TADDM launch page is made up of four sections. From left to right, top to bottom, they are:

Tivoli Application Dependency Discovery Manager (TADDM)

- The Product Console
- Download & Support
- The Administrative Console
- The Domain Manager Link

We will cover the Product Console in detail in the section “The Product Console” .The Download & Support section (top, right) contains a link to the TADDM information center, a web page with all of the TADDM documentation and other TADDM resources available online.

The Administrative console (bottom, left), is a graphical status summary of each of the TADDM servers. These are the same server components displayed with the “*./control status*” command.

The Domain Manager section (bottom, right) contains a link to the TADDM Domain Manager, which we discuss in “The TADDM Domain Manager” section of this document.

Customizing TADDM

There are three main areas used to customize and access TADDM resources. They are:

- Domain Manager
- Product Console
- Collation.Properties file

In this section, we will take a look at each of these components.

The TADDM Domain Manager

Although the TADDM documentation suggests that you should start with the Product Console after TADDM is up and running, we found that it was better to start with the Domain Manager. There were three main reasons for this:

- There is no need to configure your browser to download and run the Product Console Java application.
- With the exception of the **Administration** drawer, the Domain Manager is view only.
- The Domain Manager can create additional user ID's used to sign on to the Product Console.

Following our recommendation will enable you to become familiar with TADDM and its features without the risk of inadvertently updating or breaking your configuration or installation. However, there will be little to display because a discovery has yet to be run.

To start the Domain Manager, click the **Domain Manager** link from on the launch page. This will open a Web page with the Domain Manager's login panel, prompting for a user ID and password. When logged in, the Domain Manager's default panel is displayed. Along the left side are the drawers: **Discovery**, **Topology**, **Analytics**, and **Administration**.

Tivoli Application Dependency Discovery Manager (TADDM)

Below this is the **Discovered Components** detail area. The right side of the screen is divided into two sections. The top section is a context sensitive overview area for the task selected on the left side. The bottom section contains the details for what is selected in the overview area above. This is the same layout used in the Product Console.

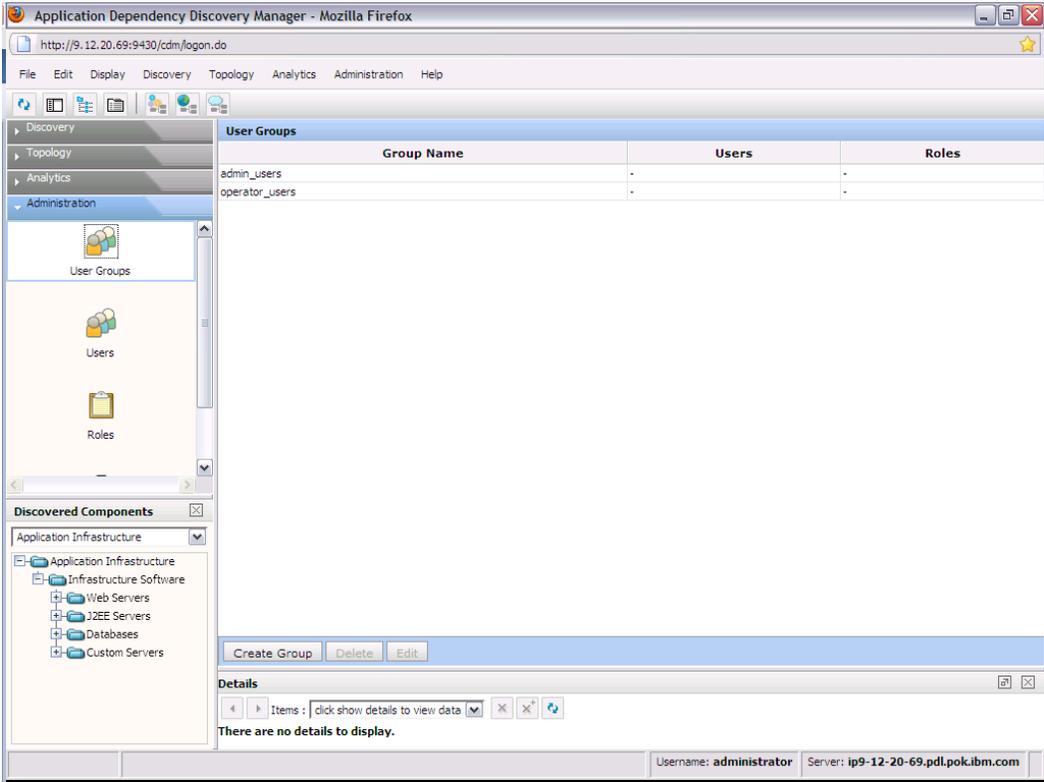


Figure 1. Example of TADDM Domain Manager

Tivoli Application Dependency Discovery Manager (TADDM)

One of the main purposes of the Domain Manager is to make security changes such as defining users and their roles.

Creating a user

Users can be assigned specific roles depending on the type of tasks that they are expected to perform. The predefined roles in TADDM are Operator, Supervisor, and Administrator. In this example we will create a user that can be used as an Administrator with the Domain Manager as well as the Product Console.

From the Domain Manager, select the **Administration** drawer on the left side of the screen. Then, select users and click **Create User** on the bottom left of the *Users* window. This brings up the Create User dialog. Because we are creating our primary administration user, assign it to the Administrator's role and give it the DefaultAccessCollection. Now log out of the Domain Manager and log back in with the user you just created.

For more information about setting up TADDM users, see chapter 13 of *Tivoli Application Dependency Discovery Manager User's Guide*, in the section "Creating a user." For more information about using the Domain Manager, see chapter 13 of the *User's Guide*, starting with the section, "Exploring the Domain Manager".

Another feature unique to the Domain Manager is the Business Intelligence and Reporting Tools (BIRT), located within the **Analytics** drawer. We will be discussing BIRT reports later in this document in the section "Viewing the results with BIRT Reports". Additional information about BIRT Reports can be found in the *Tivoli Application Dependency Discovery Manager User's Guide* in the section "Reporting with Business Intelligence and Reporting Tools (BIRT)".

We recommend that you become familiar with the interface within the Domain Manager, where you do not risk inadvertently changing anything, then when you are comfortable with the interface, start working with the Product Console. The interface is very intuitive, so it should not take long to become comfortable with it.

The TADDM Product Console

The Product Console is where most TADDM work takes place. It is the interface used to set up and run discoveries, set scopes, build access lists, create Custom Servers, review topologies, compare components, as well as perform a number of other functions. The Product Console is a Java™ application, requiring Java 2 Platform Standard Edition 5.0 to be installed on the workstation running the Web Interface.

Tivoli Application Dependency Discovery Manager (TADDM)

Before running the Product Console for the first time, make sure that you have the proper Java version running on your workstation. You can query the Java version using the **Java -version** command on the workstation, as shown in the example below.

```
C:\>java -version
java version "1.5.0"
Java 2 Runtime Environment, Standard Edition (build pwi32devifx-20060124)
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 Microsoft® Windows® XP x86-32
j9vmwi3223ifx-2006
0124 (JIT enabled)
J9VM - 20051027_03723_IHdSMR
JIT - 20051027_1437_r8
GC - 20051020_AA)
JCL - 20060120
```

For more details on preparing to run the Product Console Java application, see chapter 5 of the *TADDM Planning and Installation Guide*, in the section, “Starting the Product Console.”

Debugging problems starting the Product Console

When starting the Product Console for the first time, make sure that you enable the Java console on your workstation. This step is recommended in order to debug any issues that you might encounter while starting the Product Console (such as the one describe in “Debugging problems starting the Product Console”).

Upon initial startup of the Product Console, we experienced a hang after entering the user ID and password. Unfortunately, we had no indication as to what was happening from the Product Console itself because no errors or messages were displayed or logged. Working with level 2 support, we were instructed to enable messages on the Java console. While reviewing the messages on the Java console, we quickly became aware of an error with our network interface setup and corrected the problem. In the Java logs it was apparent that TADDM was using the wrong interface to reply to the logon. We felt it was important to share this information, because so many problems can be debugged by simply enabling messages on the Java console. Also, if you need to open a problem with the support center, the information in the Java console can be helpful in determining the resolution.

The collation.properties file

The *collation.properties* file is TADDM's main configuration file. Detailed information about many of the settings in the *collation.properties* file can be found in Chapter 8 of the *Tivoli Application Dependency Discovery Manager Administrator's Guide*. We recommend that you become familiar with this file.

A number of configuration parameters within this file are set during installation. For example, at the time of installation, the name of your database instance and associated encrypted password are set, as well as the Archive database user name. Also, the installation processes defines the TCP/IP ports used for TADDM, as well as a number of timeout values and operating system specific settings. The `collation.properties` file is located in the `$COLLATION_HOME/etc` directory, on our system it is in the `/opt/IBM/cmdmb/dist/etc` directory.

Some of the more interesting setting that you want to be aware of in the `collation.properties` file address logging properties. For example, you can set the size of a log file to a maximum of 20 MB before a rollover is performed with the setting:

```
com.collation.log.filesize=20 MB. You can set the number of rollover files retained to five with the setting: com.collation.log.filecount=5.
```

One of the more useful properties is the `com.collation.discover.engine.SplitSensorLog` setting. By setting this to true, each sensor in TADDM will create it own separate log in `$COLLATION_HOME/log/sensors/<runid>/`. Due to the fact that some of these logs are very verbose, we recommend setting this value. Logging levels are another important feature in TADDM. TADDM supports six different levels of logging: FATAL, ERROR, WARM, INFO, DEBUG and TRACE. By default, logs are set to INFO. For more detailed information about setting logging levels, see chapter 8 of *Tivoli Application Dependency Discovery Manager Administrator's Guide*, in the section “Properties for logging.”

Level-1 Discovery

A Level-1 Discovery is a credential-less discovery in which no attempt is made to log in to a remote system. A Level-1 discovery provides less detail than the other two discovery types, Level-2 and Level-3

Preparing to run a level-1 Discovery

With TADDM installed and configured, it is now time to set up a Level-1 Discovery run. Level-1 Discoveries have two prerequisites.

- The **nmap** RPM must be installed on the TADDM server.
- You can verify that nmap is installed by running the rpm query command and grep for nmap.

```
rpm -qa | grep nmap.  
nmap-4.00-14.2
```
- A scope must be defined to TADDM.
- A scope gives TADDM the IP addresses to include when running a discovery.

Creating a scope

A *scope* is simply a subnet, range, or list of IP addresses that TADDM will probe during discovery. We recommend starting with a very small scope for the first discovery. The scope can always be deleted or modified later. Scopes are the fundamental building block with respect to how TADDM approaches discoveries.

The following procedure is used to create a scope:

1. Log on to the Product Console.
2. Click the **Discovery** drawer to open it, then click the **Scope** icon.
Result: The scope overview is displayed on the right side of the screen.
3. Click **Add Set** and enter a name for the scope. We entered Test.
Result: Test is displayed in the Scope Sets.
4. In the Scope Sets, click on Test, then click **Add**.

Result: The **Add Scope** panel is displayed.

Because this is going to be a test run, we created a small range of IP addresses that contains some known servers that are active on our network, as follows:

1. Select the **Include** box and then select **Range** from the list.
2. Add a range of IP addresses and name them. We used the name TestRange.
3. Click **OK**.

Result: The range is displayed in the overview panel.

It is possible to configure a variety of different scopes for discovery in TADDM. This is useful because it is not necessary to discover every system in your environment each time that a discovery is run. In some circumstances, increasing the frequency of a system discovery might be counter productive due to workload peaks, criticality of the server, or even the dynamics of the systems configuration. TADDM provides flexibility through the setup of scopes.

Each level of TADDM discovery is addressed in more detail in the following sections.

Running a level-1 discovery

TADDM discovers active IP addresses within the scope and targets them for an nmap scan. TADDM however, might attempt a connect to a remote TELNET or SSH port to determine the operating system, but will not attempt a login. An SNMP probe can be launched to obtain more information about the resources on a system.

The following procedure is used to start a Level-1 Discovery.

1. Under the **Discovery** drawer, click the **Overview** icon.
2. Click **Run Discovery**.

Result: This brings up the Run Discovery dialog.

3. Select **Test** from the list of scopes.
4. In the selection list next to the word **profile**, select **Level 1 Discovery**
5. Click **OK**.

Result: The discovery process begins.

The **Discovery Status** resides at the bottom left of the Product Console and displays **Running** while TADDM is running a discovery. Next to the status, TADDM will also display the number of errors encountered. Also, the overview screen in TADDM will display status and details about each of the sensors running. When the discovery has completed, the status in the overview screen will change to **Idle**. To review the details of a discovery after it has completed, click **History** within the **Discovery** drawer, then select the run that you are interested in viewing. A report containing information on each run sensor will be displayed.

The initial discovery could take a while to run, depending on the scope and number of active IP addresses within the scope. You can verify that the discovery process is running by issuing the `ps -ef` command on the TADDM server and `grep` the output for `nmap`. On our system, a Level-1 Discovery running for the first time with a scope of 192.168.71.0/24, resulted in the following `nmap` request.

```
taddm@ip9-12-20-69:~> ps -ef | grep nmap
root  26711 23645  0 15:35 pts/0    00:00:18 nmap -T Normal -O -sS -sU -p
U:37,42,53,67,114,123,137,138,161,162,445,500,525,547,1718,1719,1900,5988,5989,7000,139
91,T:21,22,23,25,37,42,53,66,67,77,80,88,98,101,105,108,109,110,111,113,118,119,135,136,1
37,138,139,143,150,158,161,256,389,411,423,443,445,449,464,513,515,523,547,563,593,631,6
36,749,750,751,754,760,761,900,990,995,1008,1025,1214,1352,1433,1434,1477,1512,1521,15
33,1720,1741,1742,2000,2033,2034,2049,2119,2135,2483,2484,2809,2811,2401,2500,2638,33
00,3301,4661,4662,4665,5000,5060,5988,5989,7777,7880,8000,8007,8009,8080,8470,8471,84
72,8473,8474,8475,8476,8477,8478,8479,8880,8888,9060,9080,9090,9443,9495,11371,18554,
18555,50000 -oX - 192.168.71.195 192.168.71.35 192.168.71.32 192.168.71.33 192.168.71.30
192.168.71.31 192.168.71.150 192.168.71.114 192.168.71.80 192.168.71.137 192.168.71.138
192.168.71.251 192.168.71.110 192.168.71.142 192.168.71.108 192.168.71.248
192.168.71.106 192.168.71.48 192.168.71.105 192.168.71.145 192.168.71.49 192.168.71.202
192.168.71.201 192.168.71.104 192.168.71.144 192.168.71.249 192.168.71.186 192.168.71.44
192.168.71.45 192.168.71.185 192.168.71.42
```

Because TADDM is only looking at active TCP/IP ports, it will not discover systems that are powered down or have no IP presence on the network.

During our Level-1 Discovery, 76 items in total were discovered, of which 75 were correctly identified as: the 192.168.71.0/24 subnet, (62) Linux images, (5) Windows machines, (1) AIX®

Tivoli Application Dependency Discovery Manager (TADDM)

system and (5) IBM System i[®]. Under the heading of “Other Computer Systems” a total of five items were discovered including: a CISCO router, a controller for our tape library, a z/OS[®] instance, a z/VM[®] CMS user and an AIX network installation manager, which TADDM mistook for a host running OS/X. Considering this was all done without actually logging into any of the systems, the results are quite accurate.

Not what we expected

Most operating systems for IBM System z, z/VM and z/OS, were not discovered with a Level-1 Discovery. This is because when TADDM attempts to access these systems using TELNET or SSH, the results are returned as EBCDIC instead of ASCII. Unfortunately TADDM does not know how to interpret the EBCDIC response. At the time this paper was written, Enhancement Request MR0910095042 had been submitted requesting TADDM discover z/VM and z/OS during a Level-1 Discovery. However, in order for this request to be implemented, customer interest might be required. If you feel this is a worthwhile enhancement, express your opinion to the TADDM team.

Although z/OS is not discovered with a Level-1 Discovery, you can still have TADDM collect data on z/OS systems with a Discovery Library Adapter (DLA). DLA's are installed on the targeted operating system and send data to directly TADDM. No discovery is required.

We also experienced problems with systems that return incorrect data to TADDM. TADDM output data is only as good as the input data it receives. We have one piece of hardware that TADDM identified as a router. In this case, the device is actually Tape Library Configurator with an SNMP problem. Although IP forwarding is not actually enabled on the system, SNMP is incorrectly configured to report that it is, thus, returning the snmp IPForwarding string to TADDM. TADDM interprets any system with SNMP IPForwarding enabled as a router. Because this is not actually a TADDM issue, in this case the solution was to correct the problem on the Tape Library Configurator.

Viewing the results of the Level-1 Discovery

TADDM provides a number of ways to review the results of a discovery, and no one view is best. It all depends on what you are looking for. The most basic method is to query the database directly. However we do not recommend this method because it will provide you only with the raw data; but we do want to point out that this method is possible. Throughout this paper, we will discuss and demonstrate different methods for reviewing the results stored in the TADDM database.

One of the easiest ways to review what is stored in the TADDM database after a Level-1 Discovery is to run an inventory report. This is done by selecting the **Analytics** drawer and clicking the **Inventory** task. Select each component that you want included in the report, and click **Run Report**. The inventory report displays output for only one component type at a time. Clicking on the **Component Type** drop down menu displays each of the different types listed in the report.

Viewing the results with Business Intelligence Reporting Tools (BIRT)

TADDM includes an interface to BIRT reports. To generate an inventory BIRT report, logon to the Domain Manager, select the **Analytics** drawer and select **BIRT Reports**. This will display a list of available BIRT Reports included with TADDM. Select **TADDM_L1_DISCOVERY** from the list and press **Run Report**. On the parameters panel, accept the defaults and click **OK**. This will generate a report listing every system that was discovered in a single report. See Appendix A for an example of the TADDM_L1_DISCOVERY BIRT Report that we ran.

In addition to the BIRT reports included with TADDM, you can design and import your own BIRT reports for execution.

Level-2 Discovery

Preparing to run a level-2 discovery

Unlike a Level-1 Discovery, which requires only a scope, or range of IP addresses, a Level-2 Discovery requires both a scope and access to the operating systems at those IP addresses. Therefore, TADDM must have access to a user ID on each system that is going to be explored. TADDM refers to this user ID as the service account. In our environment, which is primarily Linux, we have set up a Linux user ID on each system with the name of taddm. This user ID is exclusively for the use of TADDM discoveries.

TADDM obtains the credentials required to logon to the service account with an **access list**. The access list contains the credentials used to ssh into the service account. Therefore, a service account must have the same password on all systems.

Service Account requirements

Each operating system will have different requirements for the service account. On Linux, TADDM requires that each service account have authorization to run the “lsof” and “netstat” commands. By default, lsof authority is reserved for root. For security reasons, it is not advised to give the service account full root authority. In fact, the service account should have just enough authority to issue the lsof command.

This can be accomplished in one of two ways:

1. setuid of the lsof command to root.
2. Set up pseudoaccess as root with the NOPASWD flag for lsof.

We chose to setuid of lsof to root with the ‘chmod 4555 /usr/bin/lsof’ command. Setting the permission bits of lsof to 4555 permits any user on the system read access and execute permissions. If you need a more granular restriction, use pseudo access for the access to lsof.

Using this approach, define a service account on each Linux system targeted for the Level-2 Discovery and configure lsof authority on each system. The next step in preparing for a Level-2 Discovery is to define a TADDM access list.

For more information about setting up service accounts for particular services or operating systems, see chapter 4, “Discovery process overview,” in *Tivoli Application Dependency Discovery Manager Administrator’s Guide*.

TADDM access lists

Access lists define the credentials required for a Level-2 Discovery. For Linux systems, this would be the user ID and password. Unlike tools that require the deployment of agents, TADDM uses sensors that reside on the TADDM server. Access lists define the credentials these sensors need to gain access to systems in your environment to further explore each system. Access lists also define whether the authentication method will be password-based or key-based.

Creating an access list for Linux system discovery

The following steps create an access list for Linux access.

1. Within the **Discovery** drawer, click the **Access List** icon.
2. In the Access List panel, Click **Add**.
Result: This brings up the Access Details dialog as shown in Figure 2.
3. On the Access Details panel, select **Computer System** for the Component Type.
4. For the name and user name we entered **taddm**
5. Enter in a Password and leave the Authentication Type as Default, Then Click **OK**.

Result: Computer System entry will be added to the Access List.

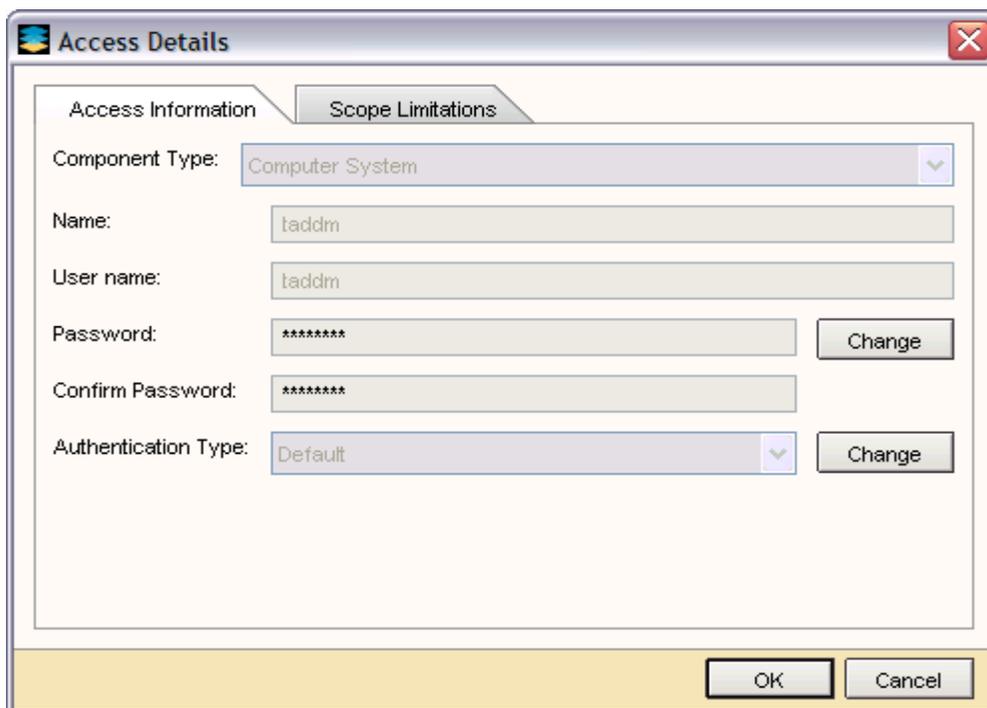


Figure 2. Example of TADDM Access Details Panel.

Running a Level-2 Discovery

To prepare for a Level-2 Discovery, a service account and access list was created. The only procedural difference between running a Level-1 and Level-2 Discovery is specifying Level-2 in Step 4 of This procedure:

1. Under the **Discovery** drawer, click the **Overview** icon.
2. Click **Run Discovery**.

Result: This brings up the Run Discovery dialog.

3. Select **Test** from the list of scopes.
4. In the selection list next to the word **profile**, select **Level 2 Discovery**
5. Click **OK**.

Result: The discovery starts to run.

Known Problems:

Novell SUSE Linux (SLES) 11 ships with Version 4.8 of the lsof command. The output from this version of lsof is a slightly different format from previous versions. This causes the output to be parsed incorrectly by TADDM. The problem has been identified and will be corrected with TADDM APAR IZ64180. As a workaround, we copied an older lsof binary from a SLES 10 SP3 system and placed it on each of our SLES 11 systems. This is **not** a supported IBM work-around, but we found it useful for our immediate needs.

Viewing results of the Level-2 Discovery

A Level-1 Discovery is essentially an inventory of the systems discovered. Although useful, it cannot tell which systems have interdependences, the services provided, or even what version of the operating system is running. A Level-2 Discovery provides these details for each system or service discovered. Without having to logon manually, administrators can obtain detailed information about operating systems, files systems, memory, network interfaces and in the case of a Linux, the installed RPM's (including version and release details). This information is stored in the TADDM database and can be viewed in some very meaningful ways.

Viewing the topology

When TADDM stores information into the CMDB database, TADDM makes no distinction between what is discovered in a Level-1, Level-2, or Level-3 Discovery. Information in the database is integrated into graphs, views and reports that give a complete picture of your environment. The topology view is just one method of graphically viewing data in the TADDM database

The Application Infrastructure Topology view, Figure 3, demonstrates how TADDM constructs dependencies from information discovered with a Level-2 Discovery. Due to the fact that TADDM can only discover dependencies for active processes, you might need to run more than one Level-2 Discovery to get a complete view of your environment. As a result of TADDM's database inventory append, it is not necessary for every service to be actively running each time that a discovery is run for TADDM, in order to continue to enhance the topology view. In addition, dependencies can be added manually if needed.

The topology view is a useful tool to determine environmental changes. TADDM retains discovery details for each run and can readily find environmental changes between two run dates. By selecting **Show Changes** under **Topology** on the menu bar and entering the date range, TADDM will highlight the changes in yellow within the topology view. Figure 3 demonstrates this feature.

Tivoli Application Dependency Discovery Manager (TADDM)

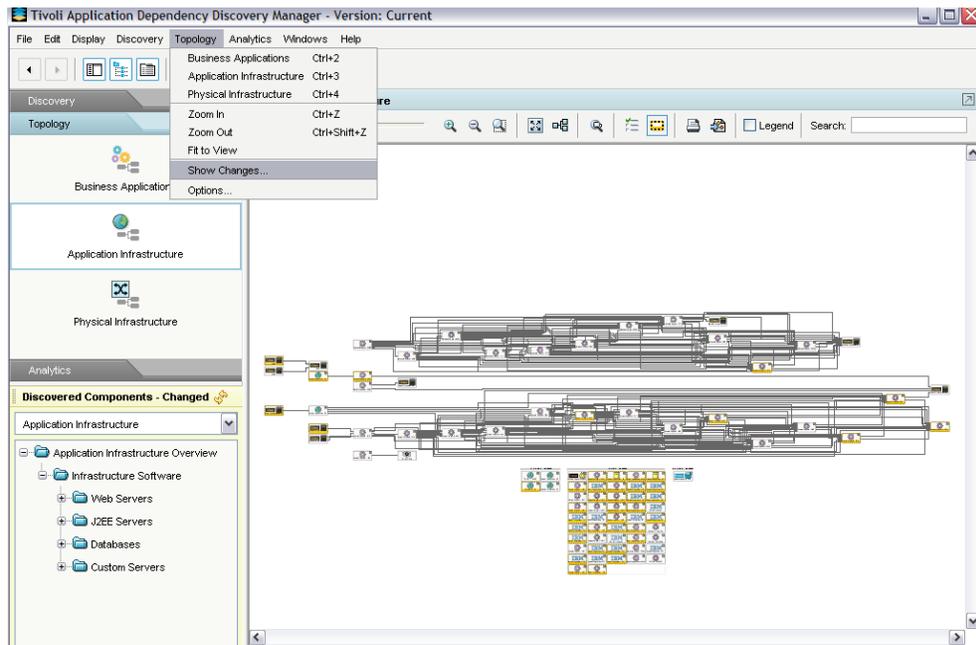


Figure 3

Custom servers are used to define additional services that are not part of the services TADDM captures by default, for TADDM to store in the database. These are services discovered by TADDM sensors, but no rules exist for adding them to the database. Therefore, these services do not appear in any of the TADDM views or reports other than the Unknown Process list.

For example, the ZENworks Management Daemon (ZMD) that runs on many SUSE Linux systems is discovered by TADDM sensors, but does not appear in any of the TADDM topology graphs, discovered components or inventory reports. If the TADDM administrators wanted to collect information about ZMD, they could do so by defining a Custom Server for this service. Note that TADDM requires a process have an associated TCP/IP service listed in netstat for the process to be discovered. TADDM provides two methods for defining Custom Servers.

The first method is to enter the criteria for which TADDM is to match, by manually defining a Custom Server. This is done by selecting the **Discovery** drawer and clicking on the **Customer Server** task. From within the Customer Server panel, click **Add** to bring up the Custom Server Details Panel. On this panel, enter the match criteria for the service to store in the database. We do not recommend this method because it is prone to entry errors and, what data you expect to create a match might not do so. For example, when we started using TADDM, we created a Custom Server to store information about a Linux service that we use to collect information about our z/VM system, named zvmprobe. We manually created the Custom Server by specifying the identifying criteria for the program name, zvmprobe, and then saved the

Tivoli Application Dependency Discovery Manager (TADDM)

definition. However, when we ran a Level-2 Discovery, nothing was stored in the database for this Custom Server. We verified that the spelling and path of process was correct in the Custom Server definition and that the service was running at the time of the discovery. We also verified that TADDM was able to discover the Linux system hosting this process. However, what we failed to check was if TADDM was capable of finding this process. Because the process we were attempting to identify did not use any TCP/IP services, TADDM was unable to locate the process.

The second method for defining a Custom Server is to use the Unknown Process List. The Unknown Process list resides within the Analytics drawer and is a complete listing of all processes that TADDM has discovered, but does not yet store in the database. It is the most reliable and recommended method for creating Custom Servers.

If we had used the Unknown Process list to define our Custom Server, we would have known TADDM was unable to discover the process we were attempting to match with. The following example (Figure 4) demonstrates the use of the Unknown Process list to define a Custom Server for the ZENworks Management Daemon (ZMD).

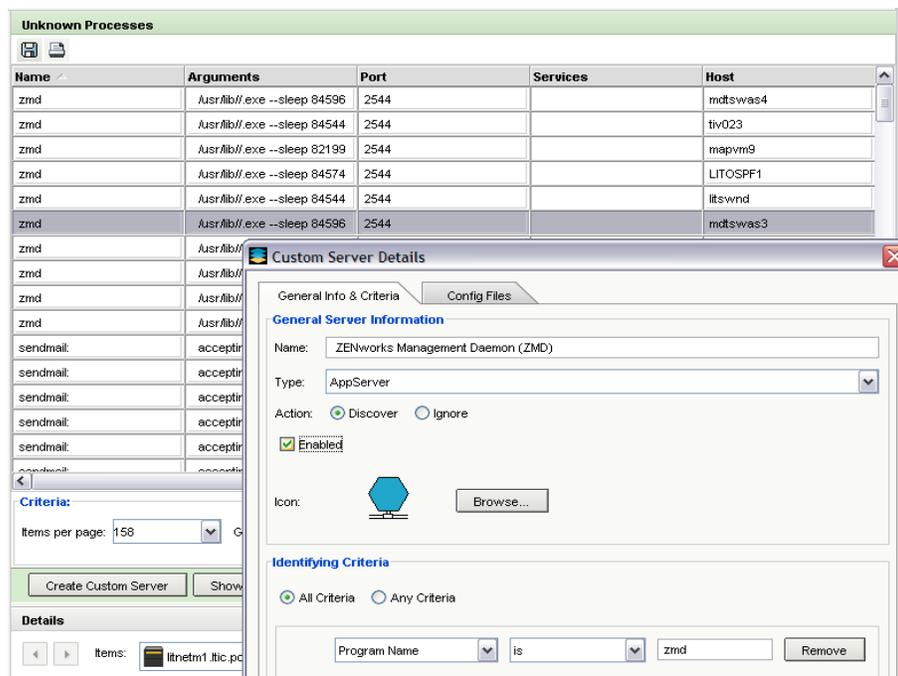


Figure 4. Example of TADDM Custom Server creation

Tivoli Application Dependency Discovery Manager (TADDM)

1. Under Analytics, select Unknown Processes.
2. Highlight the process in which you want to create a Custom Server
3. Click Create Custom Server at the bottom of the Unknown Process panel.
Result: This will bring up the Custom Server Details panel.
4. Enter a value for Name for the Custom Server or use the system provided name.
The values for Action and Identifying Criteria will be filled in by the system.
5. Check Enable so that the Custom Server will run at the next discovery.
6. Click OK in the bottom of the panel to complete the process.

You might be asking yourself why we would want to create a Custom Server for the ZENworks Management Daemon. It is not part of any of our critical application paths and we do not really need to see it in the Application Infrastructure Topologic view. In fact, it would just clutter the view if nothing else. This is exactly why we want to create a Custom Server. Having the ZENworks Management Daemon listed in the Unknown Processes view just clutters the view. As you can see in Figure 5, we have returned to the Custom Server definition for the ZENworks Management Daemon and selected **Ignore** for the action. This will prevent ZMD from appearing in the Unknown Process list and any other view within TADDM.

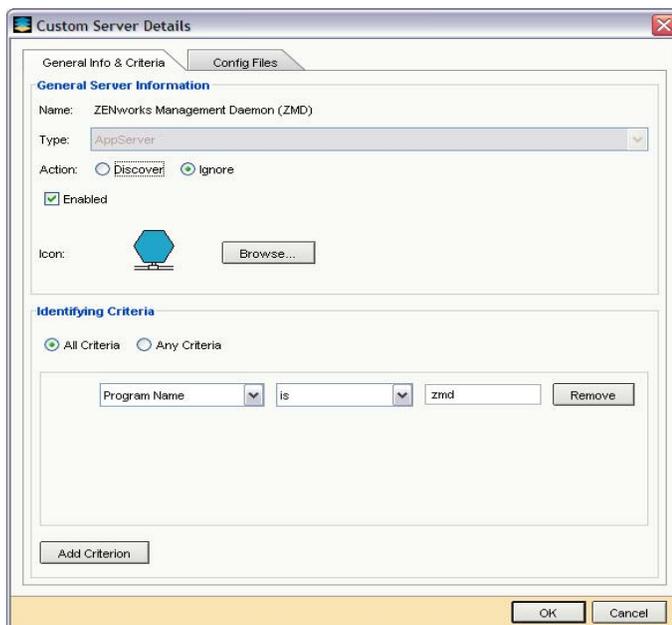


Figure
TADDM Custom Server Details.
Not working as expected

5. Example of

After specifying **Ignore** in the Custom Server properties for ZMD, it still appears in the Unknown Process list.

When TADDM ignores a process, it does not add it to the database, and it does not remove it from the Unknown Process list. It is as if the process was never discovered. The reason it remains in the Unknown Process list is because the retention period for processes in the list is set to five days by default. If you were to wait five days, the process would be removed from the list.

TADDM provides a way to change the retention period for the Unknown Process list. The retention period for Unknown Processes is specified in the `collation.properties` file with the setting `com.collation.topobuilder.RuntimeGcUnknownServerRetentionSpan= xxx`. The default value for `xxx` is 5 days. The following example of a `collations.properties` file specifies a retention period of one day for an Unknown Processes.

```
#####  
#Maximum age in days of unknown servers before they are eligible for purging  
#####  
# Maximum value is 14. A value of 0 causes all unknown servers to be purged  
# after discovery.  
#  
#com.collation.topobuilder.RuntimeGcUnknownServerRetentionSpan=5  
#  
#   Change Value to 1 per PMR 24256,180   DKB  
#  
com.collation.topobuilder.RuntimeGcUnknownServerRetentionSpan=1
```

We set the retention period to one day, because we run a scheduled discovery at 11:00 PM daily. This way, the entry does not appear in the Unknown Process list the following morning.

Note: A value of 0 will delete EVERYTHING in the Unknown Process list.

For a complete list and latest information on the sensors that TADDM provides, see the Sensor Reference section in the online TADDM Information Center.

Collecting additional data with Custom Servers

Custom servers are used to modify which data is collected by an existing sensor. In addition, you could define your own data collection rules. Custom Servers also let you define which data or configuration files you want TADDM to monitor. Such monitoring can be useful for change analysis and other analysis. For example, if you want TADDM to capture the configuration files from a domain name server, you could create a new Custom Server to do that, by completing these steps:

1. Under Discovery, select Custom Servers.
2. Click Add at the bottom of the Custom Server overview.

Result: This will bring up the Custom Server Details panel.

3. Enter a value for Name and select the type from the Type list for the Custom Server that you want to create.

We entered the name LIT_DNS_PROBE and selected the AppServer type.

4. Make sure that Discover and Enabled are selected.
5. In the Identifying Criteria section, select All Criteria.
6. For the details, select

Program Name, contains, and enter named, as shown in Figure 6.

Result: This will cause TADDM to look for the program name **named** running on a system when the next discovery is run.

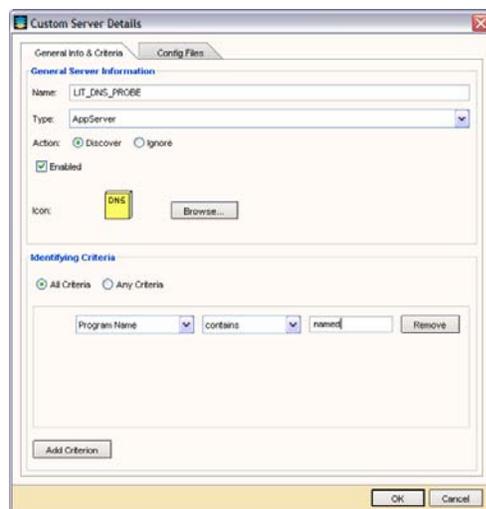


Figure 6. Example of TADDM Custom Server Details panel

Now, enter information about the configuration files you want to capture, as follows:

1. Click the **Config Files** tab on the Custom Server Details panel.
2. Click **Add**.
3. Enter the name of the file you would like to capture, as shown in Figure 7.
4. Repeat Step 3 for each file you would like to capture.

In our case, we entered the following files:

```
/etc/named.conf  
/var/lib/named/master/ltic.pok.ibm.com  
/var/lib/named/master.168.192.in-addr.arpa
```

5. Click **OK** to save the Custom Server.

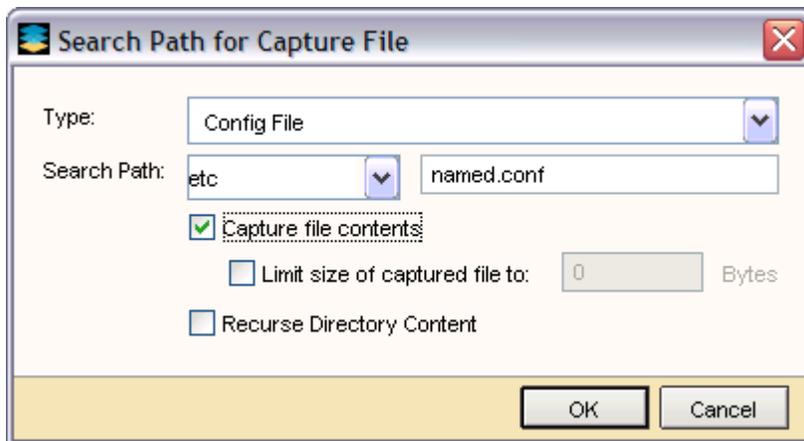


Figure 7. Example of TADDM Search Path for Capture File panel

The Custom Server is now displayed in the **Custom Server** list. You can use the **Move Up** and **Move Down** buttons to change the search order for your Custom Server in the discovery process. Custom Server templates are compared with the process list that is discovered on a computer system. If you create or modify a Custom Server to capture a file, a Level-2 or Level-3 Discovery is required. Also, an access list must be associated with the target system.

The most efficient way to review the results of a Custom Server is to use **find the service** within the Discovered Components section on the lower left side of the Product Console. Select **List/Search** in the upper drop-down menu box and then locate the Custom Server name within the **Component** drop-down. In the example shown in Figure 8, we used LIT_DNS_PROBE as our component, Right-click one of the servers listed below it, and select **Show details**. This causes TADDM to display details about the service in the **Details** panel on the lower right (Not shown in Figure 8).

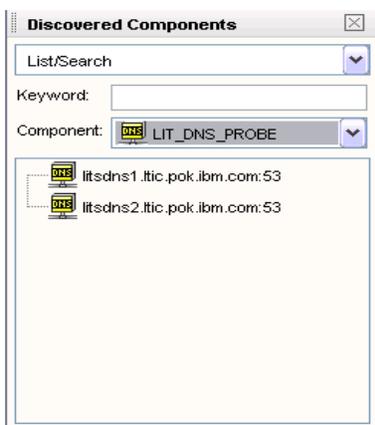


Figure 8. Example of TADDM Discovered Components.

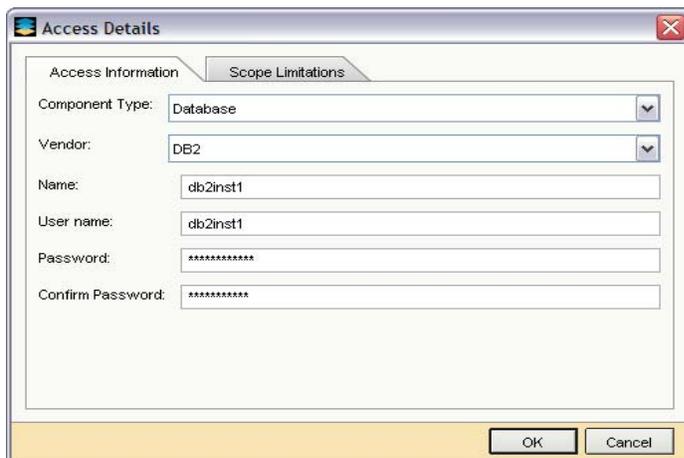
Note: If you expected to find a specific instance of a service and it is not in the list, disable the Custom Server and see if the process is in the Unknown Process list. If it is not in the Unknown Process list and you know it is using a TCP/IP service listed by netstat, check to make sure that the lsof command is installed on the target system. Also check to make sure that the version of lsof is one supported by TADDM.

Level-3 Discovery

Preparing to run a level-3 discovery

You might have noticed a lack of detail for DB2, HTTP servers, and other middleware in the topology view after the Level-2 Discovery was run. These services require deeper (Level-3) Discovery, and access lists specifically designed for them. In this section, we demonstrate how to configure TADDM to discover a DB2 instance.

For TADDM to collect data and store information about a DB2 instance, you must first create an access list designed for DB2 instances as shown in Figure 9. This is accomplished by building an access list with the **Component Type** set to *Database* and the **Vendor** set to *DB2*, and entering the db2instance name, user ID and password. In addition to the access list, you might also want to create a scope specifically for DB2, and limit this access list to that scope. This is done by selecting the **Scope Limitations** tab in the **Access Details** panel.



The screenshot shows a dialog box titled "Access Details" with two tabs: "Access Information" (selected) and "Scope Limitations". The "Access Information" tab contains the following fields:

- Component Type: Database (dropdown menu)
- Vendor: DB2 (dropdown menu)
- Name: db2inst1 (text field)
- User name: db2inst1 (text field)
- Password: ***** (password field)
- Confirm Password: ***** (password field)

At the bottom of the dialog box are "OK" and "Cancel" buttons.

Figure 9. Example of TADDM DB2 Access List Detail.

With the DB2 access list in place, it is time to run a Level-3 Discovery. For information about which credentials are required for different components types, see the online sensors section within the sensor reference on the TADDM Information Center.

Tivoli Application Dependency Discovery Manager (TADDM)

After completion of a Level-3 discovery, details of the DB2 instance are stored in the TADDM database, and are available for viewing from within the Discovered Components section of TADDM, or by clicking on the DB2 instance in one of the topology graphs.

To view details from Discovered Components, select a component type of DB2 Instance, then highlight one of the DB2 instances in the list. From within the Details panel, select the **Database** tab. To display details on the database of interest, click on the database name.

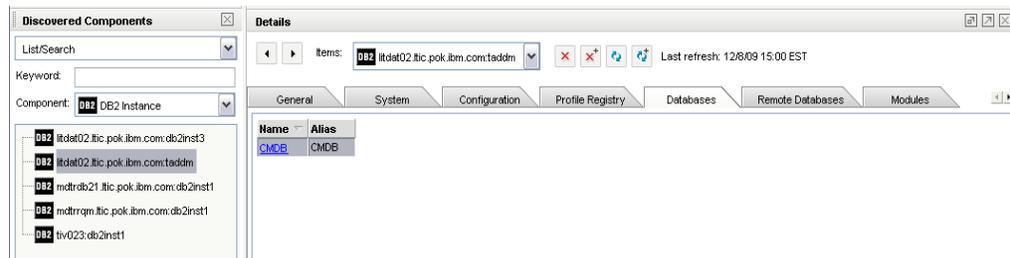


Figure 10. Example of TADDM DB2 instance details.

This will display a new set of tabs associated with the database. From this view, details about table spaces, buffer pools, configuration values and more are viewable by selecting the respective tabs. Figure 11 is an example of configuration and table space information that TADDM collected from DB2.

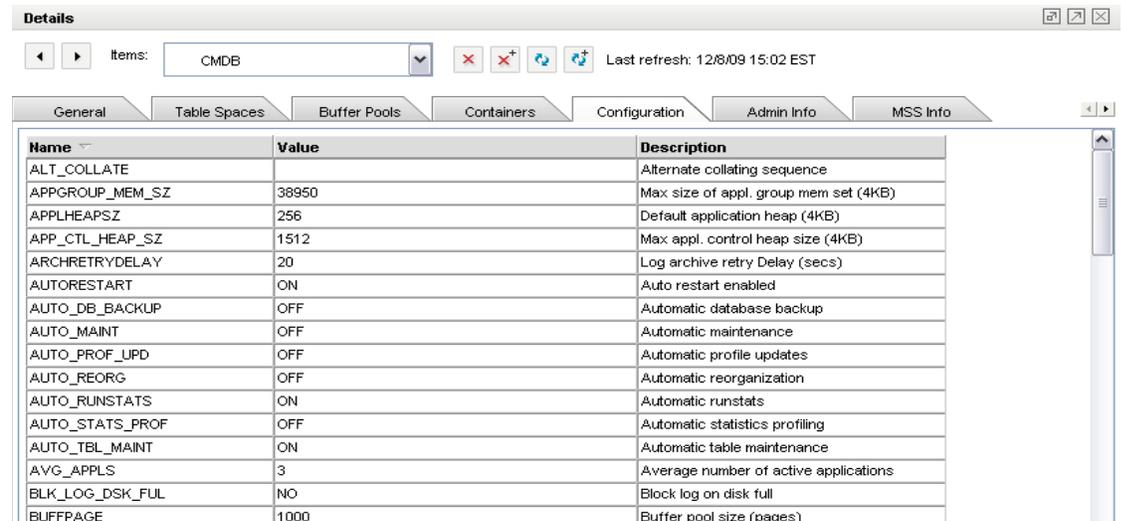


Figure 11. Example of TADDM DB2 instance and Table Spaces details.

Business applications and business services

By default, TADDM categorizes applications and services by the type of service that they provide. Some examples of different types of services include: HTTP, NFS, LDAP, DB2, DNS, routers, and operating systems. In the Topology view, TADDM builds dependencies based on network connections, and renders views based on these network relationships. However, this default topology graph-based grouping might not agree with the way we think of our infrastructures. For example, some HTTP servers could be part of an online retail business, while others are used by the HR department to view employee records. Suppose that LDAP server A is part of the retail business application used to authorize customers to their accounts, while LDAP server B is used to authenticate the identity of an HR employee and what records they can view. The default Application Infrastructure Topology will include both of these LDAP services in a single view, which might not be useful to department administrators.

The same could be true for the software administrators that support these groups. They want to view these services in groups that tightly align with the businesses they support. For example, the system administrators responsible for the retail applications could want one topology, while software administrators for HR requires a totally different logical grouping. On the other hand, the software people responsible for the individual HTTP, LDAP and DB2 servers could require a view based on server software levels. To accomplish these logical groupings, TADDM uses collections which are defined as *business applications* and *business services*.

Business Applications

Business applications are used to group servers into more meaningful business views. For example, the billing and receiving departments could have a collection of HTTP, LDAP and DB2 servers used for their business applications. Meanwhile, the Human Resources department will have a different collection of HTTP, LDAP and DB2 servers for their business applications. However, keep in mind that components used in one business application are not excluded from use in another. TADDM provides this flexibility by providing the full list of components in the database available for selection when building a Business Applications view.

Business Services

Business services are used to group servers into more meaningful “service” views. The people responsible for software updates on HTTP servers might not care if the server is part of the billing and receiving application or the human resources application; they are concerned only with the software levels of these HTTP servers. There might be a company policy that states that all HTTP servers will be at the same level. However, in the software development department, where they are creating the future versions of their software solution, they could have a business server for production HTTP servers, and a separate business service for prototype HTTP servers. No matter what the case, TADDM provides the flexibility to meet all these requirements.

Tracking software levels with TADDM

TADDM can expedite the time it takes to verify software levels in your environment by removing the need to logon and logout manually. TADDM software tracking can remove inaccuracies incurred with keeping software inventory databases current. Current and prior software levels are retained in the CMDB database, so that you can review the evolution of your environment since the inception of TADDM in to that environment. This could be important when investigating or debugging a problem in a complex application made up of many components. In some instances, it could be the only reliable method for empirical change control.

In the following example, we demonstrate how TADDM can be used to check software for Linux as well as versions of the components that make up the Linux operating system. TADDM stores the contents of the entire RPM database in the CMDB database as shown in Figures 12 and 13.

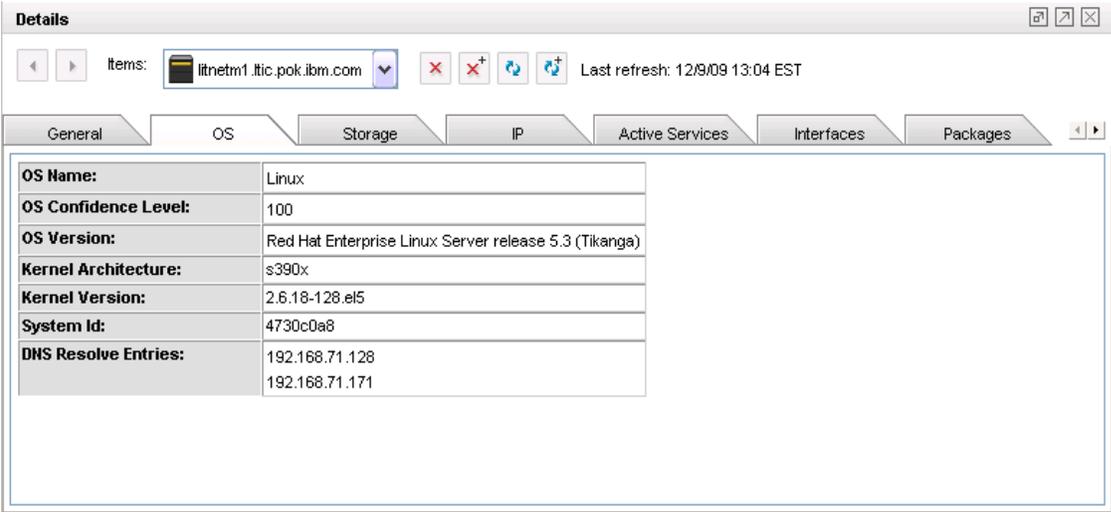


Figure 12. Example of TADDM OS tab in Details panel

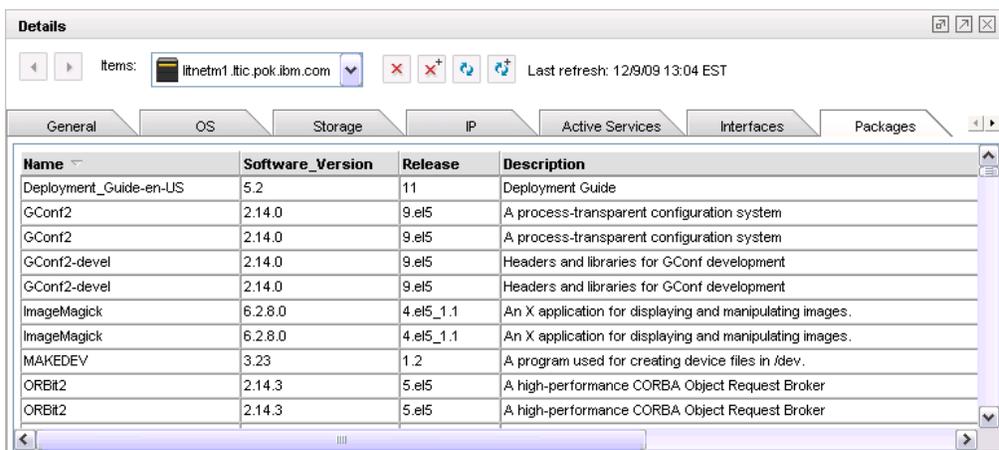


Figure 13. Example of TADDM Packages tab in Details panel

Component Comparison

The most efficient way to check software levels is to use the component comparison tool in TADDM. In our environment, we need assurance that clustered systems run the same software level. To do this, we must have at least one system with verified levels of software installed. This system will be known as the “key” system to TADDM during the component comparison process.

In the following example, we will set the host named litsha21 as the key system, and hosts named litsha22 and litsha23 as the systems for comparison. We start by clicking on the **Component Comparison** task within the **Analytics** drawer on the TADDM main panel. Then select the **Component Type** from the drop-down menu. In this example, the component type is *Linux Computer System*. Then select your key system from the **Available Components** list and click **Add**. Highlight this system in the **Included Components** list and click **Set as Key**. Now add the systems to compare to the key system by selecting them from the **Available Components** list and **Add** them to the **Included Components** list. Since we want to compare all aspects of the systems, select **Deep** for the level and then click **Run Report**. Figure 14 shows the details of our selection.

Tivoli Application Dependency Discovery Manager (TADDM)

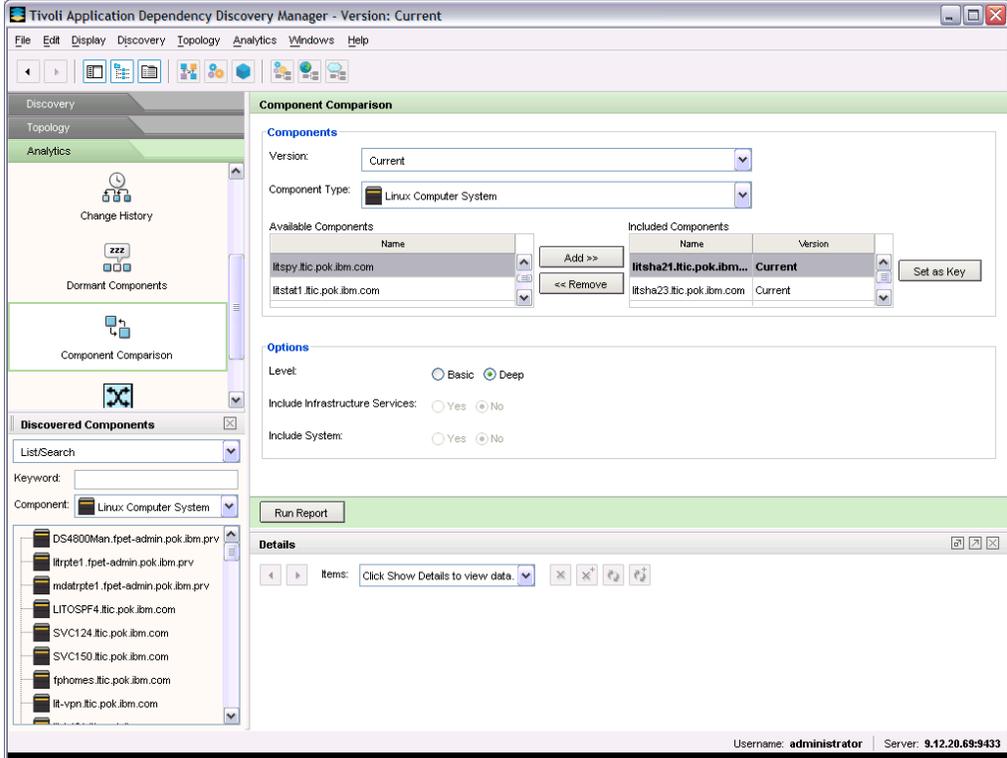


Figure 14. Example of TADDM Component Comparison setup.

Report generation time depends on the complexity of the database and how many items TADDM has to compare. Figure 15 shows a Component Comparison report. In the subset displayed in Figure 15, it is clear that litsha21 and litsha22 are good matches with the minor exception of Perl. Perl does not appear to be installed on litsha22. However, litsha23 has a number of components that are not installed on the system, noted by the [Not Set] tag in the report. Also, the level of openssl and drdb seem to be down level. It is up to the system administrator to know if these differences are acceptable or if action should be taken.

Tivoli Application Dependency Discovery Manager (TADDM)

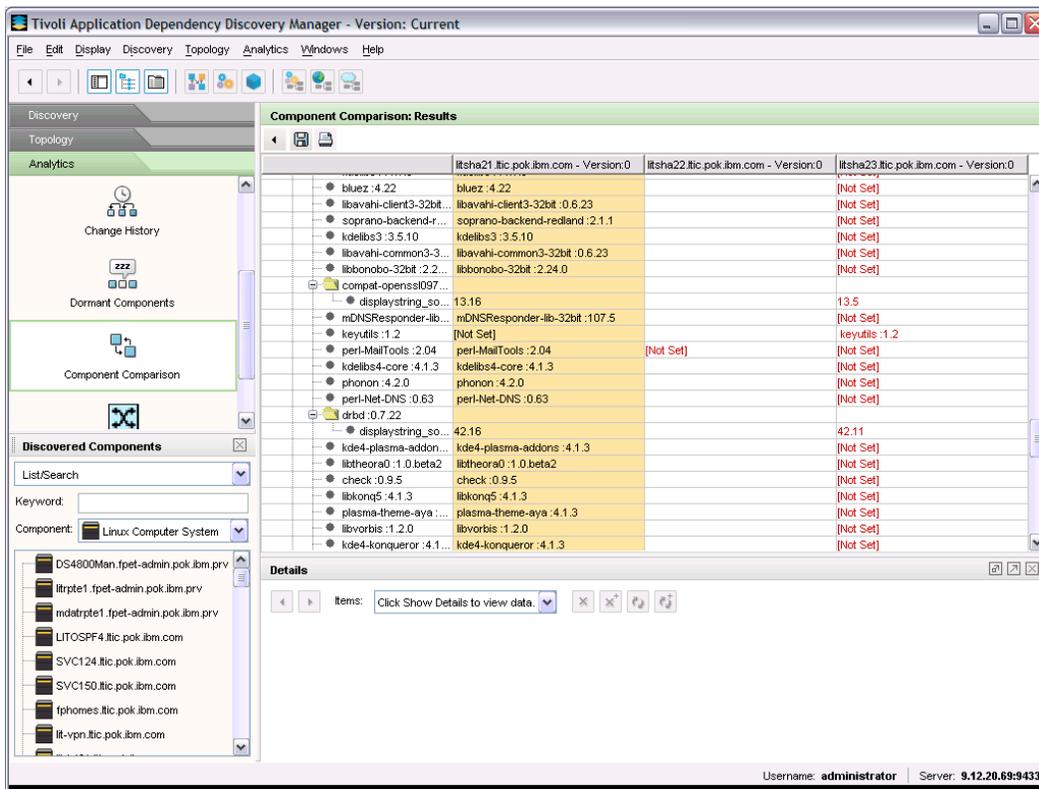


Figure 15. Example of TADDM Component Comparison report.

Summary: Experiences with TADDM in our environment

In our environment we have a number of different teams working on the same IBM System z hardware. Each team has their own set of software and server requirements. Using Business Applications and Business Services, we are able to build logical groupings that fit our needs. For example, our Integration Test team has HTTP load balancers, HTTP servers, LDAP, DB2 and Tivoli products that make up the workloads. However, our Middleware software test team also has a similar collection of HTTP, LDAP and DB2 servers. Because the mission of these teams are vastly different, they each have their own unique requirements for software levels. By logically grouping these servers into different Business Application and Business Services, it simplifies keeping track of what the different types of servers are used for.

Tivoli Application Dependency Discovery Manager (TADDM)

TADDM also provides a means for finding unused and or unneeded processes running on the hardware. For example, many of the Linux system in our environment have no need for printing, but could have the cupsd process running. TADDM provides a efficient method to locate these processes across a large number of Linux systems in a relatively short amount of time. This might not be of much concern in a distributed environment, on hardware with lots of free processor time. However, in a virtualized environment such as z/VM, resource consumption becomes cumulative so it is desirable not to waste processor time and memory on processes that are not needed. TADDM has been very helpful in identifying these processes in our environment.

Tivoli Application Dependency Discovery Manager (TADDM)

Appendix A: TADDM_L1_DISCOVERY BIRT Report

Concise Computer System Inventory Report

Report parameters

IP Address Filter IP addresses that match the pattern "192.168.71.%"

Operating System Name or Control Software Name All operating systems

Operating System Name or Control Software Name

Concise Computer System Inventory Report

IP Address	Computer System Display Name	Operating System Name or Control Software Name
192.168.71.0		
192.168.71.1	pdl-router.ltic.pok.ibm.com	Foundry IronWare
192.168.71.10	Ultrium Tape Library Specialist	
192.168.71.100	litdcon1.ltic.pok.ibm.com	Linux
192.168.71.101	litwas1.ltic.pok.ibm.com	Linux
192.168.71.102	litwas2.ltic.pok.ibm.com	Linux
192.168.71.104	litdat01.ltic.pok.ibm.com	Linux
192.168.71.105	litwas3.ltic.pok.ibm.com	Linux
192.168.71.106	litwas4.ltic.pok.ibm.com	Linux
192.168.71.108	litrsmb1.ltic.pok.ibm.com	Linux
192.168.71.109	litstems.ltic.pok.ibm.com	
192.168.71.11	SVC150.ltic.pok.ibm.com	Linux
192.168.71.110	litrlog1.ltic.pok.ibm.com	Linux
192.168.71.114	ldap01.ltic.pok.ibm.com	Linux
192.168.71.114	ldap01.ltic.pok.ibm.com	Linux
192.168.71.115	ldap02.ltic.pok.ibm.com	
192.168.71.117	litdat02.ltic.pok.ibm.com	Linux
192.168.71.119	litsteps.ltic.pok.ibm.com	Linux
192.168.71.12	SVC124.ltic.pok.ibm.com	Linux
192.168.71.120	littam01.ltic.pok.ibm.com	Linux
192.168.71.121	litstat3.ltic.pok.ibm.com	Linux
192.168.71.122	mdtldap1.ltic.pok.ibm.com	Linux
192.168.71.123	Z0.ltic.pok.ibm.com	
192.168.71.124	TPN.ltic.pok.ibm.com	
192.168.71.125	JE0.ltic.pok.ibm.com	
192.168.71.126	JC0.ltic.pok.ibm.com	IBM i5/OS® (OS/400®)
192.168.71.128	litsdns1.ltic.pok.ibm.com	Linux
192.168.71.129	litwas2.ltic.pok.ibm.com	
192.168.71.129	litwas2.ltic.pok.ibm.com	Linux
192.168.71.130	litwas3.ltic.pok.ibm.com	Linux
192.168.71.131	192.168.71.131	Linux

Tivoli Application Dependency Discovery Manager (TADDM)

192.168.71.132	mdtwas2.ltic.pok.ibm.com	Linux
192.168.71.134	litswnd.ltic.pok.ibm.com	Linux
192.168.71.136	litdcon2.ltic.pok.ibm.com	Linux
192.168.71.137	litrwas1.ltic.pok.ibm.com	
192.168.71.137	litrwas1.ltic.pok.ibm.com	Linux
192.168.71.138	litrwas4.ltic.pok.ibm.com	Linux
192.168.71.142	192.168.71.142	
192.168.71.142	mdtstsms	Linux
192.168.71.144	192.168.71.144	
192.168.71.144	mdtstsmc	Linux
192.168.71.145	litstat4.ltic.pok.ibm.com	Linux
192.168.71.148	litrdat1.ltic.pok.ibm.com	Linux
192.168.71.15	lticvm9.ltic.pok.ibm.com	IBM VM/CMS
192.168.71.150	192.168.71.150	Linux
192.168.71.157	ip9-12-20-69.pdl.pok.ibm.com	Linux
192.168.71.158	JB0.ltic.pok.ibm.com	IBM i5/OS (OS/400)
192.168.71.159	J80.ltic.pok.ibm.com	IBM OS/390® V5
192.168.71.160	JF0.ltic.pok.ibm.com	IBM i5/OS (OS/400)
192.168.71.163	192.168.71.163	
192.168.71.163	tiv023	Linux
192.168.71.164	192.168.71.164	
192.168.71.164	tiv024	Linux
192.168.71.165	192.168.71.165	
192.168.71.165	tiv025	Linux
192.168.71.166	192.168.71.166	
192.168.71.166	tiv026	Linux
192.168.71.167	tiv025	Linux
192.168.71.17	192.168.71.17	
192.168.71.17	lit10sp3	Linux
192.168.71.171	litsdns2.ltic.pok.ibm.com	Linux
192.168.71.177	litstsm.ltic.pok.ibm.com	Linux
192.168.71.185	litstat5.ltic.pok.ibm.com	Linux
192.168.71.186	litstat6.ltic.pok.ibm.com	Linux
192.168.71.192		
192.168.71.194	litrou3.ltic.pok.ibm.com	
192.168.71.194	litrou3.ltic-dmz.pok.ibm.prv	Linux
192.168.71.195	litrou4.ltic.pok.ibm.com	Linux
192.168.71.201	litsha21.ltic.pok.ibm.com	Linux
192.168.71.202	litsha22.ltic.pok.ibm.com	
192.168.71.202	litsha22.ltic.pok.ibm.com	Linux
192.168.71.203	litsha23.ltic.pok.ibm.com	Linux
192.168.71.205	litlvsvip.ltic.pok.ibm.com	

Tivoli Application Dependency Discovery Manager (TADDM)

192.168.71.208	nimmast.ltic.pok.ibm.com	AIX
192.168.71.211	pce_aix1.ltic.pok.ibm.com	AIX
192.168.71.220	litstat1.ltic.pok.ibm.com	Linux
192.168.71.220	litstat1.ltic.pok.ibm.com	Linux
192.168.71.221	DS4800Man.fpet-admin.pok.ibm.prv	Linux
192.168.71.223	ds4800-1a.ltic.pok.ibm.com	
192.168.71.224	ds4800-1b.ltic.pok.ibm.com	
192.168.71.232	192.168.71.232	
192.168.71.233	192.168.71.233	
192.168.71.236	192.168.71.236	
192.168.71.245	mapvm9.ltic.pok.ibm.com	
192.168.71.245	mapvm9.ltic.pok.ibm.com	Linux
192.168.71.248	mapvm6.ltic.pok.ibm.com	
192.168.71.248	mapvm6.ltic.pok.ibm.com	Linux
192.168.71.249	litdir00.ltic.pok.ibm.com	Linux
192.168.71.251	littrout71vip.ltic.pok.ibm.com	
192.168.71.251	littrout3.ltic-dmz.pok.ibm.prv	Linux
192.168.71.255		
192.168.71.29	lit-vpn.ltic.pok.ibm.com	Linux
192.168.71.30	litspy.ltic.pok.ibm.com	Linux
192.168.71.31	J90.ltic.pok.ibm.com	IBM i5/OS (OS/400)
192.168.71.32	JA0.ltic.pok.ibm.com	IBM i5/OS (OS/400)
192.168.71.33	sspc1.ltic.pok.ibm.com	Microsoft Windows
192.168.71.34	192.168.71.34	
192.168.71.34	LITOSPF1	Linux
192.168.71.35	192.168.71.35	
192.168.71.35	LITOSPF2	Linux
192.168.71.36	192.168.71.36	
192.168.71.36	LITOSPF3	Linux
192.168.71.37	192.168.71.37	
192.168.71.37	LITOSPF4.ltic.pok.ibm.com	Linux
192.168.71.38	litospf5.ltic.pok.ibm.com	Linux
192.168.71.39	Z1.ltic.pok.ibm.com	Microsoft Windows
192.168.71.40	ltic0016.ltic.pok.ibm.com	Linux
192.168.71.42	fphomes.ltic.pok.ibm.com	Linux
192.168.71.44	Z2.ltic.pok.ibm.com	Microsoft Windows
192.168.71.45	Z3.ltic.pok.ibm.com	Microsoft Windows
192.168.71.48	litnetm1.ltic.pok.ibm.com	Linux
192.168.71.49	litnetm2.ltic.pok.ibm.com	Linux
192.168.71.65	cisco-abr.ltic.pok.ibm.com	
192.168.71.71	mdtrihs1.ltic.pok.ibm.com	

Tivoli Application Dependency Discovery Manager (TADDM)

192.168.71.71	mdtrihs1.ltic.pok.ibm.com	Linux
192.168.71.72	mdtsihs2.ltic.pok.ibm.com	
192.168.71.72	mdtsihs2.ltic.pok.ibm.com	Linux
192.168.71.73	mdtswas1.ltic.pok.ibm.com	
192.168.71.73	mdtswas1.ltic.pok.ibm.com	Linux
192.168.71.74	mdtrwas2.ltic.pok.ibm.com	
192.168.71.74	mdtrwas2.ltic.pok.ibm.com	Linux
192.168.71.75	mdtrdb211.ltic.pok.ibm.com	
192.168.71.75	mdtrdb21.ltic.pok.ibm.com	Linux
192.168.71.76	mdtsdb222.ltic.pok.ibm.com	
192.168.71.76	mdtsdb22	Linux
192.168.71.77	mdtswasp.ltic.pok.ibm.com	Linux
192.168.71.78	Z4.ltic.pok.ibm.com	Microsoft Windows
192.168.71.79	mdtswas3.ltic.pok.ibm.com	
192.168.71.79	mdtswas3.ltic.pok.ibm.com	Linux
192.168.71.80	mdtswas4.ltic.pok.ibm.com	
192.168.71.80	mdtswas4.ltic.pok.ibm.com	Linux
192.168.71.93	192.168.71.93	
192.168.71.93	mdtrrqm.ltic.pok.ibm.com	Linux
192.168.71.94	192.168.71.94	
192.168.71.94	mdtrrqm.ltic.pok.ibm.com	Linux

This report lists discovered IP addresses. If available, data for the associated computer system and its operating system are also displayed for each discovered IP address. You can use the parameters to filter the IP address by pattern and operating system name.

12/23/2009 16:12

1 / 1

Appendix B: Collecting data from z/VM systems

zvmprobe setup example

The purpose of the `zvmprobe` Custom Server is to gather information about z/VM and store it in the CMDB database. Although the process was quite simple, it did require a little forethought. TADDM would require a mechanism that enabled it to run commands on z/VM and store the results in the CMDB database. We used the Linux `vmcp` command as the mechanism to collect data about the z/VM system. The following outlines the steps we took to collect data from z/VM

- Create a z/VM Linux guest, LITSPY, and define the TADDM service account.
- Start a unique process running on LITSPY for TADDM to discover.
- Create the Custom Server to discover the process running on LITSPY.
- Create a directive file with the same name as the Custom Server and store it in the `$COLLATION_HOME/etc/templates/commands` directory.

Creating the z/VM Linux guest, named LITSPY

The following is an example of the z/VM directory entry for LITSPY, the Linux system used to gather z/VM information. We have assigned privilege classes A through G to this guest. Doing so enables LITSPY to issue any CMS and CP commands required to gather information about the system. For security reasons, LITSPY does not share a common “Access List” profile with other Linux systems in our environment. Both the service account name and password are unique to this system.

```
USER LITSPY XXXXXXXX 1G 2G ABCDEFG
INCLUDE LITPRO
CPU 0
CPU 1
IPL CMS PARM AUTOCR
MACHINE ESA 3
OPTION APPLMON
NICDEF 0920 TYPE QDIO LAN SYSTEM PRVV71L2
MDISK 0200 FB-512 V-DISK 1024000 MR
MDISK 0201 3390 20033 10016 IT0036
```

Starting a unique process for TADDM to discover

In order for TADDM to discover LITSPY, a process using TCP/IP must be running. In addition, the process must be unique to LITSPY so that TADDM can differentiate it from the other servers. We decided to copy the binary `/usr/sbin/sshd` to `/usr/sbin/zvmprobe` on LITSPY and issue the command with the `-p 333` option, thereby starting an ssh daemon named `zvmprobe` using port 333. We verified that the process is named `zvmprobe` and is using port 333 with the `ps -ef` command, as shown in the example below.

Tivoli Application Dependency Discovery Manager (TADDM)

```
root      15551    935    0 Dec21 ?          00:00:00 sshd: root@pts/0
root      15554 15551    0 Dec21 pts/0    00:00:00 -bash
root      16592     1    0 Dec21 ?          00:00:00 /usr/sbin/zvmprobe -p
333
postfix   20093    900    0 12:02 ?          00:00:00 pickup -l -t fifo -u
root      21992    935    0 13:23 ?          00:00:00 sshd: root@pts/1
root      21995 21992    1 13:23 pts/1    00:00:00 -bash
root      22014 21995    0 13:23 pts/1    00:00:00 ps -ef
LITSPY:~ #
```

At this point, run a Level-3 Discovery, which will add /usr/bin/zvmprobe port 333 to the Unknown Process list. Because we will need to run a few Level-3 Discoveries to verify that the process is working, we created a scope containing only the IP address for LITSPY for our testing, in order to speed up the discovery process.

Creating the Custom Server named zvmprobe

Create the Custom Server by highlighting zvmprobe in the Unknown Process list, as show in Figure B1. Then click **Create Customer Server**.

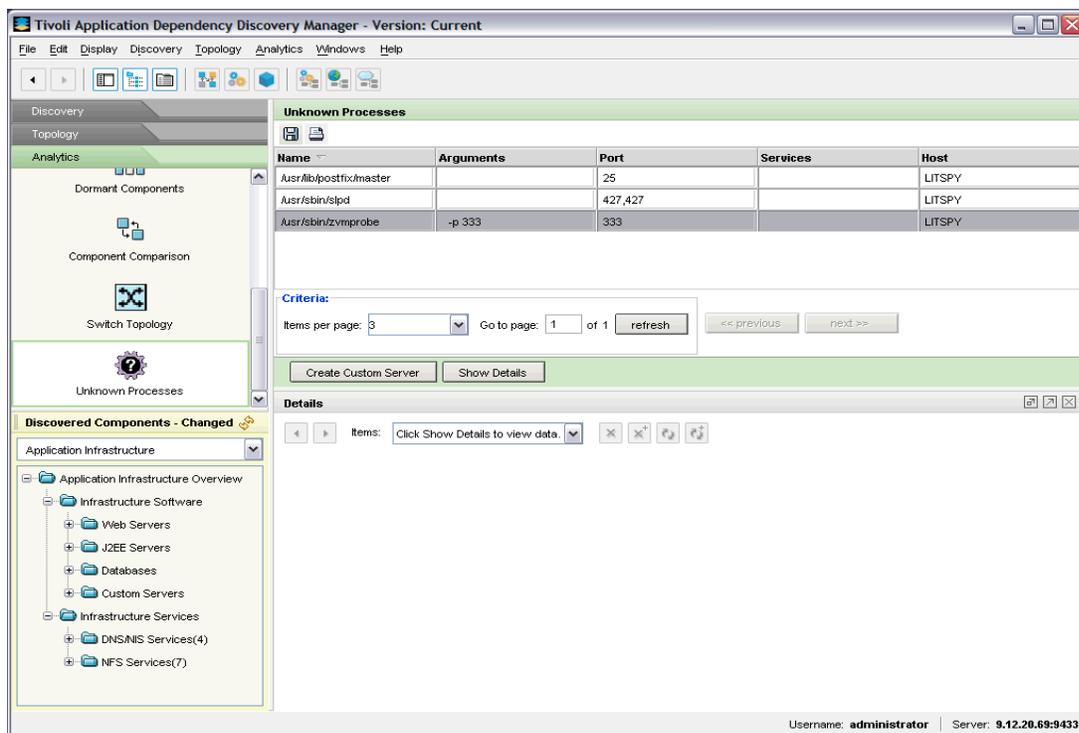


Figure B1. zvmprobe process listed in Unknown Process list.

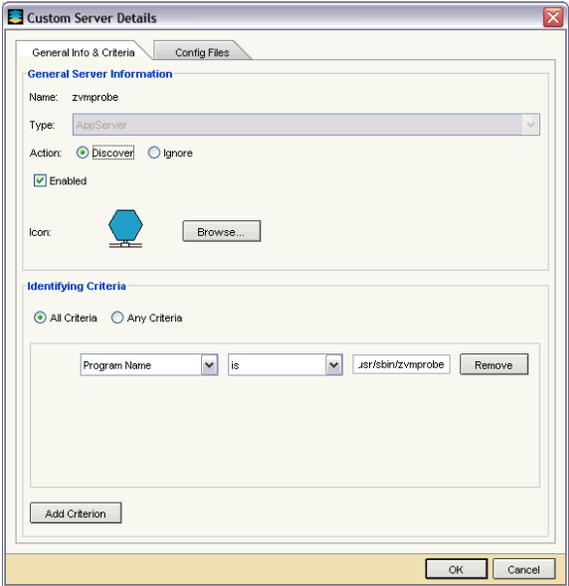


Figure B2.

Creating the zvmprobe Custom Server from the Unknown Process list.

When using the **Create Customer Server** option on the Unknown Process details panel, the name of the server will default to the name of the process. We have changed the name to zvmprobe and clicked the **“Enabled”** box under **Action** as shown in Figure B2.

Next we ran another Level-3 Discovery to verify that TADDM could find the process. The new Custom Server zvmprobe should be displayed in the Custom Servers sections of the Application Infrastructure dialog under **Discovered Components** as shown in Figure B3.

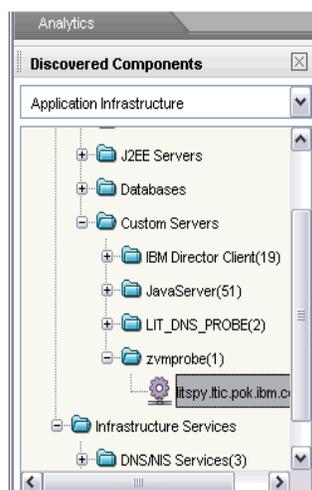


Figure B3. Displaying zvmprobe in the Discovered Components panel

Create a directive file

TADDM has a feature used to run commands on a target system and store the results in a custom configuration file. The results will be displayed in the Config Files tab on the details panel for a system as shown in figure B4. This is accomplished by placing commands in a directive file with the same name as the Custom Server in the `$COLLATION_HOME/etc/templates/commands/` directory. On our system for the zvmprobe Custom Server, the directive file would be named `/opt/IBM/cmdb/dist/etc/templates/commands/zvmprobe`, as shown in the example below.

```
Taddm:/opt/IBM/cmdb/dist/etc/templates/commands> cat zvmprobe
CMD:NOP=modprobe vmcp
CMD:CONFCONTENT.srm.data=vmcp Q SRM ALL
CMD:CONFCONTENT.vswitch.data=vmcp Q VSWITCH
CMD:CONFCONTENT.res.data=vmcp Q RES
CMD:CONFCONTENT.xstor.data=vmcp Q XSTOR
CMD:CONFCONTENT.vdisk.data=vmcp Q VDISK
CMD:CONFCONTENT.vdisk_syslim.data=vmcp Q VDISK SYSLIM
CMD:CONFCONTENT.mdc.data=vmcp Q MDC
CMD:CONFCONTENT.memassist.data=vmcp Q MEMASSIST
```

Keywords in the directive file instruct TADDM what actions to take.

A line starting with `CMD:` instructs TADDM to run the command on the target system. Following `CMD:` with `NOP=command` instructs TADDM to ignore any output resulting from command processing. Following `CMD:` with `CONFCONTENT.name=command` instructs TADDM to place the output into the Configuration Files tab in a file with the name specified.

Tivoli Application Dependency Discovery Manager (TADDM)

In the example above, CMD:NOP=modprobe vmcp instructs TADDM to load the vmcp driver on the target system. In addition, CMD:CONFCONTENT.srm.data=vmcp Q SRM ALL instructs TADDM to run the command vmcp Q SRM ALL and place the results in a file named srm.data in the Configuration Files tab.

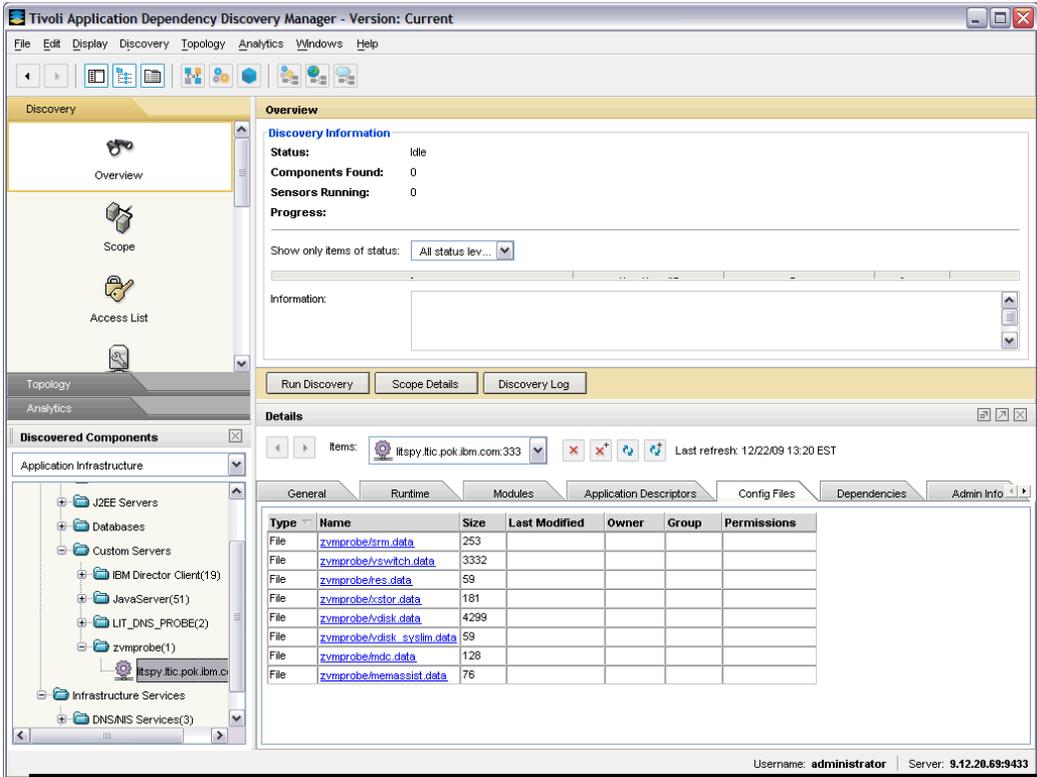


Figure B4. Config files tab on the Details panel

Details of the data stored in the resulting files is viewed by clicking on the file. Figure B5 is an example of the VSWITCH data stored in the file zvmprobe/vswitch.data.

Tivoli Application Dependency Discovery Manager (TADDM)

```
zvmprobe/vswitch.data
VSWITCH SYSTEM DT70 Type: VSWITCH Connected: 2 Maxconn: INFINITE
PERSISTENT RESTRICTED NONROUTER Accounting: OFF
VLAN Unaware
MAC address: 02-09-00-00-00-04
State: Ready
IPI timeout: 5 QueueStorage: 8
Isolation Status: OFF
RDEV: 0850.P00 VDEV: 0850 Controller: DTCVSW1
VSWITCH SYSTEM DT70TAG Type: VSWITCH Connected: 13 Maxconn: INFINITE
PERSISTENT RESTRICTED NONROUTER Accounting: OFF
VLAN Aware Default VLAN: 0001 Default Porttype: Access GVRP: Enabled
Native VLAN: 0001 VLAN Counters: OFF
IP address: 192.168.70.24 Management ID: TCPIP
MAC address: 02-09-00-00-00-07
State: Ready
IPI timeout: 5 QueueStorage: 8
Isolation Status: OFF
RDEV: 09C0.P00 VDEV: 09C0 Controller: DTCVSW1
VSWITCH SYSTEM DT71L2 Type: VSWITCH Connected: 0 Maxconn: INFINITE
PERSISTENT RESTRICTED ETHERNET Accounting: OFF
VLAN Unaware
MAC address: 02-09-00-00-00-08
State: Ready
IPI timeout: 5 QueueStorage: 8
Isolation Status: OFF
RDEV: 0854.P00 VDEV: 0854 Controller: DTCVSW1
VSWITCH SYSTEM PRVV68 Type: VSWITCH Connected: 2 Maxconn: INFINITE
PERSISTENT RESTRICTED NONROUTER Accounting: OFF
VLAN Aware Default VLAN: 0001 Default Porttype: Access GVRP: Enabled
Native VLAN: 0001 VLAN Counters: OFF
IP address: 192.168.70.25 Management ID: TCPIP
MAC address: 02-09-00-00-00-02
State: Ready
IPI timeout: 5 QueueStorage: 8
Isolation Status: OFF
```

Figure B5. Details of zvmprobe/vswitch.data file

Tivoli Application Dependency Discovery Manager (TADDM)



Copyright IBM Corporation 2010
IBM Systems and Technology Group
Route 100
Somers, New York 10589
U.S.A.

Produced in the United States of America,
01/2010
All Rights Reserved

IBM, IBM logo, AIX, DB2, i5/OS, OS/390, OS/400, System I, System z, Tivoli, WebSphere, z/OS and z/VM are trademarks or registered trademarks of the International Business Machines Corporation.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

InfiniBand and InfiniBand Trade Association are registered trademarks of the InfiniBand Trade Association. Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

ZSW03147-USEN-00