IBM Tivoli License Compliance Manager

Readme File for Fix Pack 2.3.0–TIV-TLCM–FP0002

Version 2.3

IBM Tivoli License Compliance Manager

Readme File for Fix Pack 2.3.0–TIV-TLCM–FP0002

Version 2.3

Note

Before using this information and the product it supports, read the information in “Notices” on page 63.

This edition applies to the fix pack 2.3.0–TIV-TLCM–FP0002 for version 2, release 3 of IBM Tivoli License Compliance Manager (program number 5724-D33).

IBM welcomes your comments.

Address your comments to:

IBM License Management Information Development
Rome Tivoli Lab
IBM Italia S.p.A.
Via Sciangai, 53
00144 Rome
Italy

Fax Number: (+39) 06 5966 2077

Internet ID: LMPUBS@IT.IBM.COM

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Readme File for fix pack 2.3.0–TIV-TLCM–FP0002

This readme provides important information about the fix pack 2.3.0-TIV-TLCM-FP0002 for IBM® Tivoli License Compliance Manager, version 2.3. This readme is the most current information for the fix pack and takes precedence over all other documentation.

Please review this readme thoroughly before installing or using this fix pack.

This readme includes the following topics:
- "About this fix pack"
- "Installation, migration, upgrade, and configuration information" on page 58

Note: To install the fix pack, you must be logged on as Administrator (on Windows®) or root (on UNIX®) to the computer where Tivoli License Compliance Manager is installed .

About this fix pack

Fix pack 2.3.0-TIV-TLCM-FP0002 includes support for new features and for additional agent platforms. It also includes fixes for reported APARs and defects.

This section includes the following topics:
- "New in fix pack 2.3.0–TIV-TLCM–FP0002."
- "New features" on page 2.
- "Agent platforms" on page 52
- "Support for partitioning technologies" on page 52
- "Product fixes" on page 52
- "Problems and workarounds" on page 55
- "Documentation fix history" on page 55
- "Backward compatibility" on page 57
- "Other changes as a result of this fix pack" on page 57

New in fix pack 2.3.0–TIV-TLCM–FP0002

Fix packs are cumulative, so this fix pack includes features, APARs and defects fixed included in the previous fix packs and in interim fixes for Tivoli® License Manager, version 2.3. This subsection provides a summary of the functionality and fixes that have been added for fix pack 2.3.0–TIV-TLCM–FP0002. It aims to help users who have installed the previous fix pack, 2.3.0–TIV-TLCM–FP0001 to understand what changes they can expect from this fix pack.

The following items are introduced in this fix pack:
- New features for unlicensed session management, agent configuration, complex product management, installation of GSKit, and migration of agents from version 2.1 to version 2.3.
- Fixes for APARs IY99899, IY99182, IY98781, IY98762, IY98744, IY97252, IY96542, IY96527, IY96387,

New features

The following are new features and enhancements that are installed with this fix pack.

- "Tracking unlicensed products"
- "Agent configuration" on page 7
- "Automation of complex product mapping" on page 23
- "Hardware inventory collection" on page 28
- "Agent installation size check" on page 49
- "Changes to the process for upgrading from version 2.1" on page 51.

Tracking unlicensed products

This feature introduces a new mode of tracking product instances that are installed or in use without a valid license. A product instance that does not have access to a license is referred to as an unlicensed session. Unlicensed session information is stored in the administration server database and is available for export to the Tivoli Asset Compliance Center as well as for inclusion in a report that is available from the Web UI.

Unlicensed session tracking is disabled by default. You can enable it by issuing a command from the administration server command line interface. The feature can be enabled for all monitored computers or can be restricted to specified organizations or divisions. See "Enabling tracking of unlicensed sessions."

Product instances that do not have access to a valid license are identified and added to a table of unlicensed sessions in the administration server database. See "Maintaining information about unlicensed sessions" on page 3 for information about the identification and tracking of unlicensed sessions.

A report is available on the Web UI that provides a summary or detailed view of the situation of unlicensed sessions on a specified date. The report replaces the unlicensed use report, which is no longer available after the installation of the fix pack. See"Reporting unlicensed sessions" on page 4.

When action has been taken to resolve an unlicensed situation, the database table entry can be removed. See "Clearing database entries for resolved unlicensed sessions" on page 5.

Enabling tracking of unlicensed sessions

Following installation of the fix pack, the unlicensed sessions feature is disabled by default. You can use the **unlicevent** command to enable, disable, or show the enablement status of the feature. Using the command you can enable the feature for all monitored computers and restrict enablement, disablement, or the display of the enablement status to specified organizations and divisions.

Any divisions that are created after the installation of the fix pack are created with the enablement status set to the same value as the enablement status of the organization of which they are part.

Command syntax

unlicevent (-e | -d | -s) [-recursive] [{-organization | -division} {-n <*name*> | -i <*ID*>}]

where:

-e specifies that unlicensed session monitoring is to be enabled.

-d specifies that unlicensed session monitoring is to be disabled.

-s specifies that the command is return the enablement status of the specified target.

-recursive
specifies that the enablement, disablement, or request to display the enablement status is to be propagated to the divisions of the specified organization.

-organization
specifies that the type of target for which unlicensed session monitoring is to be enabled, disabled, or shown is an organization. The organization name is specified using the **-n** parameter.

-division
specifies that the type of target for which unlicensed session monitoring is to be enabled, disabled, or shown is a division. The division name is specified using the **-n** parameter. If you have multiple organizations that include divisions with the same name, you can uniquely identify the division by specifying the ID, using the **-i** parameter.

-n *name*
specifies the name of the organization or division.

-i *ID* specifies the unique ID of the division.

Enabling unlicensed session monitoring for all organizations and divisions
To enable unlicensed session monitoring for all the computers where you have agents installed, issue the following command:

unlicevent -e

Enabling unlicensed session monitoring for selected targets
You can enable unlicensed session monitoring of selected organizations and divisions. For example, to enable monitoring of all agents in the SafeBank organization except those that are in the Testing division, issue the following commands:

unlicevent -e -recursive -organization - n SafeBank

unlicevent -d -division -n Testing

Maintaining information about unlicensed sessions

The installed software scan identifies products that are installed on each monitored computer. The scan information is reconciled with the details of product licenses and their target distributions to identify those product instances that are not covered by a currently valid license. A product instance is considered to be unlicensed if no license exists that it could use; that is, either no license exists for the product or any existing license has a target distribution defined that does not include the agent on the computer where the instance was discovered. Instances are not considered to be unlicensed if a license exists but is overused. This is considered to be an out-of-compliance situation, not an unlicensed situation.

Note: License distributions that restrict use of the license to specified application users are not considered when determining whether a product instance has access to a license. If a license exists and there is no target distribution that excludes the agent where the instance is installed, the instance is considered to be licensed even if it is not used by a specified user.

When a product instance is identified as an unlicensed session, an entry is added to the UNLICENSED_EVENT_SITUATIONS table in the administration server database. The entry identifies the product instance and records the date on which the unlicensed situation was discovered. See "Database changes" on page 6 for details of the new table.

Each time an installed software scan is received and compared with available licenses, the table is updated. Existing entries remain unchanged as long as the unlicensed situations are still present. When a product instance that has an entry in the table is no longer included in the update of unlicensed sessions, the unlicensed situation is considered to have ended and the table entry is updated with an end date. Entries that have end dates set can be removed from the table using the purgedatabase command from the administration server command line. See "Clearing database entries for resolved unlicensed sessions" on page 5.

Reporting unlicensed sessions

The unlicensed session feature introduces a new report on the Web UI. The unlicensed sessions report provides a view of the product instances that are identified as having been in an unlicensed state on a specified date. The report consists of summary and detail views. The summary view provides an overview of the severity of the unlicensed situation at product, product version, or product release level. From the summary view, you can select a product and view the details of each agent where an unlicensed session was discovered.

A product instance is considered to be an unlicensed session if at the date when it is discovered by the installed software scan and no valid license is available for it.

You can resolve an unlicensed situation in the following ways:

- If no license exists for the product or the license that was assigned to it has expired, check that you have a valid entitlement and if so create a new Tivoli License Compliance Manager license and make it available to the targets where the unlicensed instances are installed.
- If a license exists but its target distribution excludes the targets where the unlicensed sessions are installed, consider whether the license can be redistributed.
- If the product does not require a license, use the Define Product Properties task to disable discovery of the product by the installed software scan.
- If the product should not be in use on the computer where it was discovered as unlicensed, uninstall it.

To review details of software products installed without a valid license, perform the following steps:

1. In the portfolio, click **Produce Reports** and then click **Unlicensed Sessions** in the expanded task group.
2. Define the report parameters as follows:

   Hierarchy level

   Select the level of product hierarchy at which information is to be reported.

   Software vendor

   Select **All** to include products from all vendors or select one or more vendors.

   SW platform

   Select **All** to include unlicensed sessions for all operating systems or select one or more operating systems.

You can select UNIX to include products installed on any supported UNIX operating system or you can select a specific UNIX operating system, for example AIX®.

Product name
: To limit the products in the report using product name matching, type a part of a product name preceded, followed, or enclosed by wildcard characters (%). All products that have names that match are included in the report.

End date
: Specify the date up to which information about unlicensed sessions is to be included in the report. The default value shown is the latest date that can be specified because it is the date on which the database table of unlicensed sessions was last updated.

Report type
: Select the report type you require. If you select **Summary**, the report includes only the product, version or release level overview. If you select **Full details**, from the product summary you are able select a product and view details of the agents where unlicensed sessions were discovered.

3. Click **Browse report** to view the report immediately, or click **Export data** to generate the report using batch processing.

   If you selected **Browse report** , the unlicensed sessions summary is displayed, showing a table of the products for which at least one unlicensed session was discovered.

   Each row of the table shows the product, at product, version or release hierarchy level, the product vendor, the number of unlicensed sessions discovered for the product and a colored symbol that indicates the severity of the unlicensed situation of the product, as follows:

   Red (critical)
   : At least 66% of the installed instances of the product are unlicensed.

   Orange (medium)
   : At least 34% (but less than 66%) of the installed instances of the product are unlicensed.

   Yellow (low)
   : Less than 34% of the installed instances of the product are unlicensed

   Note: The report refers to unlicensed agents, but the number reported is the number of unlicensed sessions.

   You can export this information in XML format by clicking **Export data**

4. If you selected the Full details report type, you can select a product from the list and click **View details**.

   The list of agents where an unlicensed session of the selected product was discovered is displayed, showing the name of the agent, the division, the date on which the unlicensed session was discovered, and the date of the last installed software scan performed.

   You can export this information in XML format by clicking **Export data**

Clearing database entries for resolved unlicensed sessions

When an unlicensed session is resolved, an end date is set for the database table entry and the session is not included in any report that is requested with an end date that is later than the end date of the unlicensed session table entry. Entries for resolved unlicensed sessions can be cleared from the UNLICENSED_EVENT_SITUATIONS table using the **purgedatbase** command.

Note: This is an existing command, used for clearing historical database tables of entries that are older than a configurable number of months. The new command syntax not only introduces the option to remove the entries from the UNLICENSED_EVENT_SITUATIONS table but also allows you to specify number of months as a parameter of the command.

Command syntax

purgedatabase [f] {-all | -h | -u} -a <*number of months***>**

where:

f
If you specify this option, no confirmation is required before starting the cleanup action.

-all
specifies that both historical and unlicensed session data is to be cleared.

-h
specifies that historical data is to be cleared.

-u
specifies that unlicensed session data is to be cleared.

-a <*number of months***>**
specifies the minimum age of entries to be cleared.

Clearing resolved unlicensed session data

To clear unlicensed sessions that were resolved at least six months ago, , issue the following command:

purgedatabase -u -a 6

Database changes

The unlicensed sessions feature includes the following changes to the administration server database:

- Addition of the UNLICENSED_EVENT_SITUATIONS table. See “The UNLICENSED_EVENT_SITUATIONS table.”
- Changes to the CUSTOMER and DIVISION tables. See “Changes to existing tables” on page 7.
- Removal of the ADM.INV_H_PROD and ADM.INV_H_DIV tables, which were previously used to store data for the unlicensed use report.

The UNLICENSED_EVENT_SITUATIONS table: Stores information about unlicensed sessions that were discovered on monitored computers.

Columns	Description	Type	Null
AGENT_ID	Unique identifier of the agent where the unlicensed session was discovered.	bigint	no
PRODUCT_ID	Unique identifier of the unlicensed product.	bigint	no
CUSTOMER_ID	Unique identifier of the organization to which the agent belongs.	bigint	no
START_DATE	The date when the unlicensed session was discovered.	date	no
END_DATE	The date when the unlicensed session was resolved. While the session remains unresolved, the end date is set to 9999-12-31, representing infinity.	date	no

Changes to existing tables: The CUSTOMER and DIVISION table are changed to include a setting that indicates the enablement status of the unlicensed session feature for each organization and division. The following field is added to both tables:

Columns	Description	Type	Null
UNLICENSED_FEATURE	Indicates whether the unlicensed session monitoring is enabled for the organization or division. Possible values are 0 (not enabled) and 1 (enabled).	smallint	no

Agent configuration

This feature introduces the means to manage changes to agent configuration using a set of commands to be issued from the administration server command line. It provides the following capabilities:

- To set agent parameters at organization, division, or individual agent level.

  When a value for a parameter is set at a higher level of the topology, it is inherited by lower levels that do not have specific values set. For example, if you set a value for a parameter at organization level, all agents in all divisions of that organization will use that value unless a different value has been set for a division or for individual agents.

  If you want the new value that you have applied at a higher level to apply to lower levels that have their own values set, you can choose to remove or suspend the values that are set at the lower levels. Values that have been suspended can later be reinstated.
- To schedule agent self-update to be performed in a specified timeslot.
- To suspend or activate defined values for agent parameters at organization, division, or individual agent level.

  The state of the a defined parameter can be set to active or hold. By controlling the state of parameters you can prepare an agent configuration ahead of time, putting each parameter on hold until the time comes to activate the new configuration.
- To view details of the parameter values applied at organization, division, or individual agent level.

Configuration changes that you make using the commands are stored in the administration server database and are downloaded to the runtime servers and then to agents in the same way as other information defined on the administration server, for example, licenses. Take into account the time required for download services between the administration server and the runtime servers and between runtime servers and agents when defining configuration changes, in particular date settings that are in the immediate future.

Note: The capability to make configuration changes at individual agent level adds a useful element of flexibility. However, be careful to not overuse this capability. Most changes should be defined at organization or division level. If you consistently define settings for individual agents, the administration server database table that stores the agent configuration information will become very large and could impact performance when topology changes are downloaded from the administration server to runtime servers.

For more details about the agent configuration management feature and example of how it can be used, see:

- “Summary of agent configuration commands”
- “Summary of agent parameters”
- “Scenario 1: Scheduling the agent self-update service” on page 10
- “Configuring a periodic agent self-update” on page 11
- “Scenario 2: Implementing and removing a test configuration” on page 11
- “Agent configuration command reference” on page 12
- “Database changes” on page 18

Summary of agent configuration commands

Table 1 lists the commands introduced for agent configuration management feature.

Table 1. Agent configuration commands

Command	Description
setconfkeyvalue	Sets the value and, optionally, the state of an agent configuration parameter.
setconfkeyvaluestate	Sets the state of the value of an agent configuration parameter.
getconf	Retrieves information about the agent configuration parameters.
getconfkeymetainfo	Retrieves meta information for a specified version of the Tivoli License Compliance Manager agent.
delconfkeyvalue	Deletes the value of a configuration parameter for a specified topology node.

See “Agent configuration command reference” on page 12 for the syntax definitions of these commands.

Summary of agent parameters

Table 2 shows the agent settings that can be managed using the agent configuration commands.

Table 2. Agent configuration parameters

	Units	**Default**	**Minimum**	**Maximum**
Parameter	**Description**			
`inv_start_date`	date	The date of inclusion in the database		
	The date and time when the first or only occurrence of the software inventory scan is performed. The format is YYYY-MM-DD.hh.mm.			
`inv_rate_type`	Integer	0	0	3
	Indicates the software scan repeating period. Possible values are: **0** No repetition **1** The repetition period is 1 day. **2** The repetition period is 7 days. **3** The repetition period is 30 days.			
`inv_rate_value`	Integer	1	1	9999
	The number of repeating periods, as defined by `inv_rate_type`, that separate consecutive occurrences of the software scan.			

Table 2. Agent configuration parameters (continued)

Parameter	Units	Default	Minimum	Maximum
	Description			
`update_enabled`	Integer	0 (disabled)	0	2
	Indicates the status of the agent self-update service. Possible values are: **0** Disabled **1** Periodic: Agents check for a new version of the agent at regular periods defined by the parameter `update_period`. **2** Scheduled: Agents a new version of the agent during a period of time defined by the start date specified by the `update_start` parameter and the length of the update period defined by the `update_interval` parameter.			
`update_period`	minutes	10080 (1 week)	1440 (1 day)	129600 (3 months)
	The interval between checks for the presence of a new version of the agent on the runtime server when `update_enabled` is set to **1**.			
`update_start`	Date	The date of inclusion in the database		
	The date and time and time at which the agent scheduled self-update time window starts if the `update_enabled` parameter is set to **2** (scheduled). Self-update is available from this date and time for the number of hours specified for the `update_interval` parameter. The format is YYYY-MM-DD-hh.mm			
`update_interval`	Hours	6	1	24
	The length of time for which the agent scheduled self-update remains open if the `update_enabled` parameter is set to **2** (scheduled). Self-update is available from the date and time specified by the `update_start` parameter.			
`ping_period`	minutes	60	60	360 (6 hours)
	The length of time the agent waits between checks of the connection to its runtime server.			
`down_parms_period`	minutes	360 (6 hours)	180 (3 hours)	10080 (1 week)
	The interval between downloads of the agent parameters from the runtime server. In addition to the parameters, at each download the agent checks the date of the last catalog update at the runtime server, and also downloads the catalog if its own catalog is older.			
`upload_usage_period`	minutes	360	360	10080 (1 week)
	The interval between uploads to the runtime server of offline software use data.			
`proc_list_period`	seconds	300	60	600
	The interval between compilation of consecutive versions of the agent process list.			
`sys_update_period`	minutes	30	30	10080 (1 week)
	The interval at which the agent performs updates of hardware information about the node or partition where it is running.			
`was_check_period`	minutes	1440 (1 day)	120 (2 hours)	2880 (2 days)
	The interval at which the agent checks to ensure that the WebSphere® Application Server agent is running.			
`upload_unk_period`	minutes	180 (3 hours)	180 (3 hours)	1440 (24 hours)
	The interval between uploads of unknown file information from the agent to the runtime server. The upload service is only run if there is information on unknown files to upload.			

Table 2. Agent configuration parameters (continued)

Parameter	Units	Default	Minimum	Maximum
	Description			
`hw_scan_enabled`	Boolean	no	no	yes
	Indicates whether the hardware inventory scan feature is enabled.			
`virtual_hw_scan_disabled`	Boolean	yes	no	yes
	For agents running in virtualized environments, indicates whether the hardware inventory scan is disabled.			
`hw_inv_start_date`	Date	The date of inclusion in the database		
	The date and time when the first or only occurrence of the hardware inventory scan is performed. The format is YYYY-MM-DD-hh.mm			
`hw_inv_rate_type`	Integer	0	0	3
	Indicates the hardware scan repeating period. Possible values are: **0** No repetition; **1** The repetition period is 1 day.; **2** The repetition period is 7 days.; **3** The repetition period is 30 days.			
`hw_inv_rate_value`	Integer	1	1	9999
	The number of repeating periods, as defined by `hw_inv_rate_type`, that separate consecutive occurrences of the hardware scan.			
`resend_times`	Integer	5	1	30
	The number of times the upload to the runtime server is reattempted after an upload operation fails.			

Note: The agent parameters that you can manage using the agent configuration management feature include the parameters that control the scheduling of software and hardware inventory scans. The principal means of the scheduling software scanning is by using the Web UI task. The principal means of scheduling hardware scanning is by using the hwscanconf command from the administration server command line. You can change the scan-related parameters using the agent configuration management commands, but for consistency with the scan scheduling methods you cannot make changes at individual agent level.

Scenario 1: Scheduling the agent self-update service

With the introduction of the new agent parameters `update_start` and `update_interval`, you can define a time window during which the agent self-update can be performed. After agents have been upgraded to version 2.3 with fix pack 2, they are able to identify the time window that has been set for updates and contact the runtime server during that period.

Like the other agent parameters, the agent self-update settings can be applied at organization, division, or agent level. This provides more flexibility, allowing you to plan a staged upgrade of groups of agents and to ensure that the update processing takes place at a time that is convenient to you.

Note: Agents that are at a version earlier than version 2.3 with fix pack 2 cannot be updated using this method because they do not recognize the time window. These agents continue to use the previous self-update logic by which when self-update is enabled the agents must check for updates at regular intervals defined by the `update_period` parameter. To update agents that are at earlier versions (for example, after installing fix pack 2), use the procedure described in "Configuring a periodic agent self-update."

To schedule the update of agents in the Sales division to take place between 22:00 on 10th July 2007 and 6.00 on the 11th July 2007, do the following:

1. Issue the following command to enable self-update for agents in the Sales division.

   setconfkeyvalue -division -n Sales -k update_enabled -v 2 -s active
2. Issue the following command to start the update period at 22:00 on 10th July 2007.

   setconfkeyvalue -division -n Sales -k update_start -v 2007-07-10-22.00 -s active
3. Issue the following command to end the update period at 6.00 on the 11th July 2007, by setting the update period to 8 hours.

   setconfkeyvalue -division -n Sales -k update_interval -v 8 -s active

Note: If you have multiple organizations that include divisions with the same name, a message is shown listing the divisions that have the specified name with their unique IDs. Reissue the command using the **-i** parameter and specifying the ID.

Configuring a periodic agent self-update

For agents at a version earlier than 2.3 with fix pack 2, configure the periodic update option. When this option is enabled, agents check the runtime server for updates at regular intervals defined by the `update_period` parameter. The default value for this parameter is 10080 minutes (1 week).

To configure the periodic update option for the agents in the Sales divisions, issue the following command:

setconfkeyvalue -division -n Sales -k update_enabled -v 1 -s active

Scenario 2: Implementing and removing a test configuration

In this scenario, a new set of values are to assigned to the parameters that control the timing of agent to runtime server communications. The values are set at for the Sales division and override values already set for individual agents by temporarily suspending the agent-level settings. The configuration is prepared in advance, put on hold and then later activated. Complete the following steps to perform these tasks:

1. Issue the following commands to define the new configuration values, put them on hold and suspend the current agent level settings.

   setconfkeyvalue -division -n Sales -k upload_usage_period -v 360 -s hold -h

   setconfkeyvalue -division -n Sales -k upload_unk_period -v 360 -s hold -h

setconfkeyvalue -division -n Sales -k down_parms_period -v 360 -s hold -h

2. Issue the following command to activate the settings.

   setconfkeyvaluestate -division -n Sales -s active

   setconfkeyvaluestate -division -n Sales -k upload_unk_period -s active

   setconfkeyvaluestate -division -n Sales -k down_parms_period -s active

To reinstate the original agent settings, issue the following commands.

setconfkeyvaluestate -division -n Sales -s active -p

setconfkeyvaluestate -division -n Sales -k upload_unk_period -s active -p

setconfkeyvaluestate -division -n Sales -k down_parms_period -s active -p

These commands send the active status to all the agents in the division for which settings were put on hold using the setconfkeyvalue parameter in the **setconfkeyvalue** command.

Agent configuration command reference

This section provides the full syntax rules for all agent configuration commands.

setconfkeyvalue:

Sets the value and, optionally, the state of an agent configuration parameter.

Syntax

setconfkeyvalue { -division | -organization } { -n *<name>* | **-i** *<identifier>* }
-k *<key>* \
-v *<value>* **[-s { active | hold }] [-r | -h]**

setconfkeyvalue -agent { -n *<name>* | **-i** *<identifier>* } **-k** *<parameter_name>* \
-v *<value>* **[-s { active | hold }]**

Description

This command sets the value and, optionally, the state of an agent configuration parameter for a specified node of the topology. If the node is an organization or a division, the **-r** and **-h** options can be used to cancel or set the value of the configuration parameter for all nodes that belong to the organization or division.

Options

-agent | -division | -organization
: Required. It indicates the topology node on which the command must have effect.

-n *<name>* | **-i** *<identifier>*
: Required. The name or identifier of the topology node. The options are as follows:

  -n *<name>*
  : The name of the topology node. If the node is an agent, name is

the agent hostname. If more than one agent has the same hostname, a warning message displays the identifiers of the agents whose hostname matches. When this happens, you must enter the command again specifying the **-i** option instead.

-i *<identifier>*
The identifier of the topology node.

-k *<parameter_name>*
Required. The name of the agent configuration parameter.

-v *<value>*
Required. The value that you define for the specified agent configuration parameter.

-s { active | hold }
Optional. Defines the state of the value of an agent configuration parameter. The mutually exclusive values are:

active When the state of the value of a parameter is set to active, the value of the parameter is effectively used.

hold When the state of the value of a parameter is set to hold, the value of the parameter is kept in the administration server database and is not used.

-r | -h Optional. These mutually exclusive options are available only if the node is organization or division. When **-h** is used the value of the configuration parameter is propagated to all agents that belong to that node. When **-r** is used the value of the configuration parameter is cancelled for all agents that belong to that node.

Authorization

Windows
A user with administrator rights.

UNIX root

Examples

- Example 1.

  To set the value of the ping_period parameter to 300 for agent lab238275, enter the command as follows:

  ```
  ITLM Admin Server-CLI>setconfkeyvalue -agent -n lab238275 -k ping_period
   -v 300
  ```

- Example 2.

  To set the value of the ping_period parameter to 300, and make its state active, for all agents that belong to division abc, enter the command as follows:

  ```
  ITLM Admin Server-CLI>setconfkeyvalue -division -n abc -k ping_period
   -s active -v 300 -h
  ```

See Also

"setconfkeyvaluestate" on page 14

"getconf" on page 15

"getconfkeymetainfo" on page 16

"delconfkeyvalue" on page 17

setconfkeyvaluestate:

Sets the state of the value of an agent configuration parameter.

Syntax

setconfkeyvaluestate [**-k** <*parameter_name*>] [**-s** { **active** | **hold** }]

setconfkeyvaluestate { **-division** | **-organization** } { **-n** <*name*> |
-i <*identifier*> } **-k** <*parameter_name*> **-s** { **active** | **hold** }
[**-p**]

setconfkeyvaluestate **-agent** { **-n** <*name*> | **-i** <*identifier*> }
-k <*parameter_name*> **-s** { **active** | **hold** }

Description

This command sets the state of the value of an agent configuration parameter for a specified node of the topology, for the topology nodes that belong to a specified organization or division, or for an entire topology.

Options

-agent | **-division** | **-organization**
: Optional. It indicates the topology node on which the command must have effect. If not used, the command has effect on the entire topology.

-n <*name*> | **-i** <*identifier*>
: Required if the topology node is specified. The name or identifier of the topology node. The options are as follows:

  -n <*name*>
  : The name of the topology node. If the node is an agent, name is the agent hostname. If more than one agent has the same hostname, a warning message displays the identifiers of the agents whose hostname matches. When this happens, you must enter the command again specifying the **-i** option instead.

  -i <*identifier*>
  : The identifier of the topology node.

-k <*parameter_name*>
: Required if the topology node is specified. The name of the agent configuration parameter.

-s { **active** | **hold** }
: Required. Defines the state of the value of an agent configuration parameter. The mutually exclusive values are:

  active When the state of the value of a parameter is set to active, the value of the parameter is effectively used.

  hold When the state of the value of a parameter is set to hold, the value of the parameter is kept in the administration server database and is not used.

-p Optional, it can only be used when the topology node is organization or division. If used, the command has effect on all nodes that belong to the specified organization or division.

Authorization

Windows
A user with administrator rights.

UNIX root

Examples

- Example 1.

  To set the state of the value of the ping_period parameter to active for all agents of division abc, enter the command as follows:

  ```
  ITLM Admin Server-CLI>setconfkeyvaluestate -division -n abc -k ping_period
   -s active -p
  ```
- Example 2.

  To set the state of the value of the resend_times parameter to hold for the entire topology, enter the command as follows:

  ```
  ITLM Admin Server-CLI>setconfkeyvaluestate -k resend_times -s hold
  ```

See Also

"setconfkeyvalue" on page 12

"getconf"

"getconfkeymetainfo" on page 16

"delconfkeyvalue" on page 17

getconf:

Retrieves information about the agent configuration parameters.

You can run this command only from the administration server command line.

Syntax

getconf -<*entity*> **{ -n**<*name*> | **-i** <*identifier*> } **[-k** <*parameter_name*>]

Description

This command retrieves information about the configuration parameters defined for a specified agent or for all nodes that belong to a division or an organization.

Options

-<*entity*>
Required. It indicates the topology node on which the command must have effect. The options are as follows: **-agent**, **-division**, or **-organization**.

-n <*name*> | **-i** <*identifier*>
Required. The name or identifier of the topology node. The options are as follows:

-n <*name*>
The name of the topology node. If the node is an agent, name is the agent hostname. If more than one agent has the same hostname, a warning message displays the identifiers of the agents

whose hostname matches. When this happens, you must enter the command again specifying the **-i** option instead.

-i *<identifier>*
: The identifier of the topology node.

-k *<parameter_name>*
: Optional. The name of the agent configuration parameter, if the command must retrieve the value of a particular parameter.

Authorization

Windows
: A user with administrator rights.

UNIX root

Examples

- Example 1.

  To retrieve the values of all configuration parameters for all agents that belong to division abc, enter the command as follows:

  ```
  ITLM Admin Server-CLI>getconf -division -n abc
  ```
- Example 2.

  To retrieve the value of the was_check_period configuration parameter for agent lab232756, enter the command as follows:

  ```
  ITLM Admin Server-CLI>getconf -agent -n lab232756 -k was_check_period
  ```

See Also

“setconfkeyvalue” on page 12

“setconfkeyvaluestate” on page 14

“getconfkeymetainfo”

“delconfkeyvalue” on page 17

getconfkeymetainfo:

Retrieves meta information for a specified version of the Tivoli License Compliance Manager agent.

You can run this command only from the administration server command line.

Syntax

getconfkeymetainfo [**-a***<agent_version>*] [**-k** *<parameter_name>* | **-d**]

Description

This command retrieves meta information for a specified supported version of the Tivoli License Compliance Manager agent and, optionally, for a specified configuration parameter. Supported agent versions are 2.3, 2.2, and 2.1. When **-d** is used, the command retrieves the details of all configuration parameters.

Meta information of a configuration parameter includes the type of parameter, the default value, and the range of possible values.

Options

-a <*agent_version*>
: Optional. It indicates the version of the Tivoli License Compliance Manager agent. Possible values are 2.3, 2.2, and 2.1. If this option is not specified, the command retrieves the meta information for all supported agent versions.

-k <*parameter_name*> **|** **-d**
: Optional. Mutually exclusive options, as follows:

  -k <*parameter_name*>
  : The name of the configuration parameter whose meta information the command must retrieve.

  -d
  : If this option is used, the command retrieves the details of all configuration parameters.

Authorization

Windows
: A user with administrator rights.

UNIX
: root

Examples

- Example 1.

  To retrieve the meta information of the hw_scan_enabled configuration parameter for Tivoli License Compliance Manager 2.3 agents, enter the command as follows:

  ```
  ITLM Admin Server-CLI>getconfkeymetainfo -a 2.3 -k hw_scan_enabled
  ```

See Also

"setconfkeyvalue" on page 12

"setconfkeyvaluestate" on page 14

"getconf" on page 15

"delconfkeyvalue"

delconfkeyvalue:

Deletes the value of a configuration parameter for a specified topology node.

You can run this command only from the administration server command line.

Syntax

delconfkeyvalue { -division | -organization } { -n <*name*> | **-i** <*identifier*> } **-k** <*parameter_name*> **[-p]**

delconfkeyvalue -agent { -n <*name*> | **-i** <*identifier*> } **-k** <*parameter_name*>

Description

This command deletes the value of a configuration parameter for a specified node of the topology. If the node is an organization or a division, the **-p** option can be used to delete the value of the configuration parameter for all nodes that belong to the organization or division.

Options

-agent | -division | -organization
: Required. It indicates the topology node on which the command must have effect.

-n *<name>* | **-i** *<identifier>*
: Required. The name or identifier of the topology node. The options are as follows:

  -n *<name>*
  : The name of the topology node. If the node is an agent, name is the agent hostname. If more than one agent has the same hostname, a warning message displays the identifiers of the agents whose hostname matches. When this happens, you must enter the command again specifying the **-i** option instead.

  -i *<identifier>*
  : The identifier of the topology node.

-k *<parameter_name>*
: Required. The name of the agent configuration parameter.

-p
: Optional. If used, the command deletes the value of the configuration parameter for all nodes that belong to the organization or division

Examples

- Example 1.

  To delete the value of the ping_period parameter for division abc, enter the command as follows:

  ```
  ITLM Admin Server-CLI>delconfkeyvalue -division -n abc -k ping_period
  ```
- Example 2.

  To delete the value of the hw_scan_enabled parameter for all node that belong to division abc, enter the command as follows:

  ```
  ITLM Admin Server-CLI>delconfkeyvalue -division -n abc -k hw_scan_enabled -p
  ```

See Also

"setconfkeyvalue" on page 12

"setconfkeyvaluestate" on page 14

"getconf" on page 15

"getconfkeymetainfo" on page 16

Database changes

The agent configuration feature adds the following tables to the administration server database:

- "The CONF_KEY table" on page 19.

- "The CONF_KEY_VALUE table."
- "The GROUP table" on page 20.
- "The GROUP_GROUP_REL table" on page 20.
- "The RESOURCE table" on page 21.
- "The RESTYPE table" on page 21.
- "The RES_GROUP_REL table" on page 21.
- "The RES_CONF_KEY_REL table" on page 22.

Changes are also made to the ADM.DIVISON and ADM.DIVISION_DELETED tables. See "Changes to the ADM.DIVISION and ADM.DIVISION_DELETED tables" on page 23.

The CONF_KEY table: Stores meta-information for each agent configuration parameter. When the fix pack is installed, this table is created and populated with an entry for each of the agent configuration parameters listed in Table 2 on page 8.

Columns	Description	Type	Null
NAME	The name of the configuration parameter	varchar (50)	no
HIER_ID	Identifies the highest hierarchy node for this implementation.	smallint	no
IS_INHERITABLE	Indicates whether the value can be inherited from higher topology nodes. Possible values are: **0** No **1** Yes	smallint	no
MAX_VISIBILITY_HEIGHT	Indicates the maximum number of levels in the hierarchy by which a value for the configuration parameter can be inherited.	smallint	no
ID	Uniquely identifies the configuration parameter.	bigint	no
PREDEFINED_VALUE	The value of this parameter, set at installation time.	varchar (50)	no
ONLY_INHERITABLE	Indicates whether the value for the configuration parameter must set for the highest hierarchy node and inherited by all underlying nodes. **0** No **1** Yes	smallint	no
VALUE_TYPE	Indicates the unit of measurement for the configuration parameter. Possible values are: **0** integer **1** string **2** timestamp	smallint	no

The CONF_KEY_VALUE table: Stores all configuration parameter values that have been defined for topology nodes. .

Columns	Description	Type	Null
TIMESTAMP	The date and time when the configuration was defined.	timestamp	no
VALUE	The value assigned to the configuration parameter.	varchar (50)	yes
STATE	Indicates whether the configuration value is in the active or hold state. Possible values are: **0** hold **1** active	smallint	no
ID	Uniquely identifies the configuration parameter value setting.	smallint	no
CONF_KEY_ID	Uniquely identifies the configuration parameter.	bigint	no
GROUP_ID	Uniquely identifies the topology node for which the configuration parameter value is defined.	bigint	yes
RESOURCE_ID	Uniquely identifies the resource with which the configuration parameter value is associated.	bigint	yes

The GROUP table: Stores information about the topology nodes for which configuration parameters can be defined.

Columns	Description	Type	Null
LEVEL	The hierarchical level of the topology node.	integer	no
GROUP_ID	Uniquely identifies the topology node.	bigint	no
HIER_ID	Identifies the hierarchy to which the topology node belongs.	smallint	no
START_TIME	Specifies the start of the period of validity of the topology node.	timestamp	no
END_TIME	Specifies the end of the period of validity of the topology node.	timestamp	yes
ID	Uniquely identifies the topology node.	bigint	no

The GROUP_GROUP_REL table: Stores information about the relationships between topology nodes for which configuration parameters can be defined.

Columns	Description	Type	Null
CHILD_ID	Identifies a topology node that has a child relation to another node.	bigint	no
START_TIME	Specifies the start of the period of validity of the parent-child relationship.	timestamp	no
END_TIME	Specifies the end of the period of validity of the parent-child relationship.	timestamp	yes
PARENT_ID	Identifies the topology node that is the parent of the node identified by the CHILD_ID.	bigint	no

Columns	Description	Type	Null
PARENT_HIER_ID	Identifies the hierarchy to which the parent node belongs.	bigint	no

The RESOURCE table: Stores information about monitored resources. Resources can be nodes, shared pools, LPARs, virtual machines, and operating system images.

Columns	Description	Type	Null
TYPE	Identifies the type of resource. Possible values are: **0** Node **1** Shared pool **2** LPAR **3** Virtual Machine **4** Operating system image	integer	no
START_TIME	Specifies the start of the period of validity of the resource.	timestamp	no
END_TIME	Specifies the end of the period of validity of the resource.	timestamp	yes
MAP_NAME	An external representation of the resource entity.	varchar (100)	yes
VERSION	Identifies a specific set of configuration parameters applied to the resource.	varchar (50)	yes

The RESTYPE table: Stores information the possible resource types.

Columns	Description	Type	Null
TYPE	Identifies the type of resource. Possible values are: **0** Node **1** Shared pool **2** LPAR **3** Virtual Machine **4** Operating system image	integer	no
CANCONSUME	Indicates whether software use data can be collected for resources of this type. Possible values are **true** and **false**.	char (32)	no

The RES_GROUP_REL table: Stores information relationships between topology nodes and resources.

Columns	Description	Type	Null
RESOURCE_ID	Uniquely identifies a resource.	bigint	no

Columns	Description	Type	Null
GROUP_ID	Uniquely identifies a topology node.	bigint	no
START_TIME	Specifies the start of the period of validity of the relationship between the topology node and the resource.	timestamp	no
END_TIME	Specifies the end of the period of validity of the relationship between the topology node and the resource.	timestamp	yes
HIER_ID	Identifies the hierarchy to which the topology node belongs.	bigint	no

The RES_RES_REL table: Stores information relationships between resources.

Columns	Description	Type	Null
CHILD_ID	Identifies a resource that has a child relation to another resource.	bigint	no
START_TIME	Specifies the start of the period of validity of the parent-child relationship.	timestamp	no
END_TIME	Specifies the end of the period of validity of the parent-child relationship.	timestamp	yes
PARENT_ID	Identifies the resource that is the parent of the resource identified by the CHILD_ID.	bigint	no
VERSION	Identifies a specific set of configuration parameters applied to the resource.	varchar (50)	no

The RES_CONF_KEY_REL table: Stores information relationships between configuration parameters and resources.

Columns	Description	Type	Null
TYPE	Identifies the type of resource. Possible values are: **0** Node **1** Shared pool **2** LPAR **3** Virtual Machine **4** Operating system image	bigint	no
CONF_KEY_ID	Identifies the configuration key	bigint	no
VERSION	Identifies a specific set of configuration parameters applied to the resource.	varchar (50)	no
VALUE_RANGE_MAX	Specifies the maximum value that can be assigned to the configuration parameter.	varchar (50)	yes
VALUE_RANGE_MIN	Specifies the minimum value that can be assigned to the configuration parameter.	varchar (50)	yes

Changes to the ADM.DIVISION and ADM.DIVISION_DELETED tables: The following fields, present in both the ADM.DIVISION and ADM.DIVISION_DELETED tables, are removed:

- INV_START_DATE
- INV_RATE_TYPE
- INV_RATE_VALUE
- SELFUPDATE

If you installed fix pack 2.3.0-TIV-TLCM-FP0001, the following fields related to the hardware inventory scan feature are also removed from both tables:

- HWINV_START_DATE
- HWINV_RATE_TYPE
- HWINV_RATE_VALUE
- HWINV_VIRTUAL_DISABLED

Automation of complex product mapping

This feature partially automates the mapping of complex products. It consists of two alternative solutions that you can apply to ease the task of mapping components of complex products :

- A command that activates the MIN_PATH mapping rule by which shared components are automatically mapped to the original, standalone product to which they belong. See "Using the MIN_PATH rule."
- A set of scripts and a Java™ standalone application to create default mappings that create mappings based on the rule that if an unshared component of a complex product is present on a monitored system, any shared components of that product that are discovered on the same system are treated as if they belong to that product. This method is referred to as the co-location rule. See "Using the Co-location rule" on page 26.

Using the MIN_PATH rule

The MIN_PATH mapping rule assumes that the most likely mapping for a shared component is to the original, standalone product to which it belongs. For example, a component of DB2® UDB Enterprise Server Edition (ESE) is potentially part of the DB2 UDB ESE product and of many other products that bundle DB2 UDB ESE. When the MIN_PATH rule is applied, any unmapped instances of a DB2 UDB ESE component that are detected are automatically mapped to the DB2 UDB ESE product. If you have an instance of the shared component that is part of a different product, you must change the mappings, using the Manage Complex Products tasks. Existing mappings are not affected by the activation of the feature.

You can apply the rule at organization level, by issuing the new **mapcomps** command. See "Managing mapping rules."

Additional information about complex product mappings has been added to the Installs Snapshot report. See "Mapping information in the Installs Snapshot report" on page 24.

New tables have been added to the administration server database to support the introduction of mapping rules. See "Database changes" on page 25.

Managing mapping rules: The **mapcomps** command is added to the administration server command line interface. Using this command you can perform the following tasks:

- Activate the mapping rule.
- Deactivate the mapping rule.
- Show the current mapping rule status

Command syntax

mapcomps {-s | -e | -d} [-organization {-n <*name*> | -i<*ID*>] [-r <*rule_name*>]

where:

-s shows the status of applied rules.

-e enables mapping rules.

-d disables mapping rules.

-organization specifies that the type of target for the command is an organization. The organization name is specified using the **-n** parameter.

-n *name* specifies the name of the organization.

-i *ID* specifies the ID of the organization.

-r *rule_name* is the name of the rule to be enabled, disabled, or shown. The default value is MIN_PATH.

Enabling the MIN_PATH mapping rule

To enable the MIN_PATH mapping rule for the SafeBank organization, issue the following command:

mapcomps -e -organization -n SafeBank -r MIN_PATH

Disabling the MIN_PATH mapping rule

To disable the MIN_PATH mapping rule for the SafeBank organization, issue the following command:

mapcomps -d -organization -n SafeBank- r MIN_PATH

Viewing the MIN_PATH mapping rule

To view the mapping rules that are applied to the SafeBank organization, issue the following command:

mapcomps -s -organization -n SafeBank

Mapping information in the Installs Snapshot report: Additional information about the mapping status of complex products and their components is included in the agents view of the Installs Snapshot report. When producing the report in the agent view, you can limit it to include only agents where unmapped components have been discovered, only agents where automatically mapped components have been discovered, or both.

An additional column, **Mapping status** (possible values, **Unmapped**, **Mapped**, and **Automatic Mapping**), and an additional button, **Complex product details**, are added to the agent view.

Selecting an agent and clicking **Complex product details** displays the details of complex products discovered on the agent in a tree structure. You can navigate the tree, expanding and collapsing its branches. Components that are mapped are displayed in bold and when you select one, its details are displayed, including how the mapping was created:

- For components mapped using one of the Manage Complex Product tasks, the details include the Web UI user ID that identifies the user who performed the task.

- For components, mapped by applying the mapping rule, the details include the name of the rule (MIN_PATH).

If the components discovered by the agent on a monitored system indicate more than one possible mapping for a shared component, the component is mapped to a product according to the rule applied, but it also shown as a possible part of one or more other products. These entries appear in grey and cannot be selected. For example, the DB2 UDB Data Warehouse product includes the components DB2 UDB Data Warehouse, DB2 UDB ESE, and WebSphere Application Server. DB2 UDB ESE and WebSphere Application Server are shared components, and application of the MIN_PATH rule maps them to the products DB2 UDB ESE and WebSphere Application Server respectively. However, if the agent also discovers the DB2 UDB Data Warehouse component, it is possible that each of these shared components is really part of the DB2 UDB Data Warehouse product. .

Database changes: The complex product mapping feature adds the following tables to the administration server database:

- SW_INV_ALGORITHM which stores the identification information for mapping rules. See "The SW_INV_ALGORITHM table"
- ALGORITHM_RELATIONSHIP which stores the relationships between each mapping rule and each entity (organization) defined in the database. See "The ALGORITHM_RELATIONSHIP table."
- BASE_PROPERTY which stores two types of information: activation or deactivation of mapping rules for an entity; definition of component links either by the automatic server process controlled by the mapping rules or by user intervention. See "The BASE_PROPERTY table" on page 26

The SW_INV_ALGORITHM table: Stores the identification information for mapping rules. Currently the only rule available is MIN_PATH

Columns	Description	Type	Null
ID	Unique identifier of the mapping rule.	bigint	no
Name	Name of the mapping rule.	varchar	no
DESCRIPTION	Description of the mapping rule	varchar	no

The ALGORITHM_RELATIONSHIP table: Stores the relationships between each mapping rule and each entity (organization) defined in the database.

Table 3. The ALGORITHM_RELATIONSHIP table

Columns	Description	Type	Null
SW_INV_ ALGORITHM_ID	Unique identifier of a mapping rule defined in the SW_INV_ALGORITHM table.	bigint	no
TYPE	Indicates the relationship between the entity (organization) and the mapping rule: **1** Applied **2** Not applied .	smallint	no

Table 3. The ALGORITHM_RELATIONSHIP table (continued)

Columns	Description	Type	Null
TARGET_TYPE	Type of entity for which the mapping relationship is defined: **1** organization **2** division (not currently used)	smallint	no
TARGET_ID	The unique ID of the organization.	bigint	no

The BASE_PROPERTY table: Stores information about the activation or deactivation of mapping rules and information about the definition of component to product links.

Columns	Description	Type	Null
ID	Unique identifier of the property	bigint	no
TYPE	Identifies the type of property: **1** Server (automatic) defined link **2** User-defined link **100** Enablement/disablement of mapping rules	smallint	no
OWNER_TYPE	Identifies the table that this property refers to: **1** Link: The row is related to information in the LINK table about links between components and products. **2** Sw_Inv_Algorithm: Not currently used. **3** Algorithm_Relationship: The row relates to information in the ALGORITHM_RELATIONSHIP table about the relationship between organizations and mapping rules.	smallint	no
OWNER_ID	Identifies the row in the LINK or ALGORITHM_RELATIONSHIP table to which this row relates.	bigint	no
VALUE_INT	For properties of TYPE 100, this identifies the mapping enablement setting: **0** Off **1** On	smallint	yes
VALUE_STRING	For server or user-defined links, identifies the user that defined the link. For server defined links, the value is "Server", for user-defined links it is a Web UI user ID.	varchar	yes
LAST_MODIFIED		timestamp	no

Using the Co-location rule

This complex product mapping rule is based on the fact that many complex product definitions include a non-shared component. This component can be used to identify the product that is installed and any shared components of that product that are installed on the same system can be assumed to be installed as part of the bundle. For example, DB2 Data Warehouse has the unshared component, DB2 Data Warehouse and the shared component DB2 UDB. If the Data Warehouse component is discovered, it is reasonable to think that the DB2 UDB component installed on the same system is a part of the DB2 Data Warehouse and not of any of the other complex products that include the DB2 UDB component.

This logic is applied to the administration server database by running a pair of SQL scripts and a Java standalone application that gather information about the complex product components installed on each deployed agent and use that information to create links based on co-location. The scripts and Java application are copied under the Tivoli License Compliance Manager installation directory when the fix pack is applied to the administration server database component.

This complex product mapping rule can only be applied to a subset of all complex products that are defined in the catalog. It is available for the following products:

- DB2 UDB Enterprise Server Edition, version 8.2
- DB2 UDB Enterprise Server Edition, version 8.1
- DB2 Data Warehouse Base Edition, version 9.1
- DB2 Data Warehouse Enterprise Edition, version 9.1
- Tivoli License Compliance Manager, version 2.2
- WebSphere Application Server Network Deployment, version 6 .1
- WebSphere Application Server Network Deployment, version 6.0
- WebSphere Application Server Network Deployment, version 5.1
- WebSphere Application Server - Base, version 6.1
- WebSphere Application Server - Base, version 6.0
- WebSphere Application Server - Base, version 5.1
- WebSphere Application Server - Base, version 5.0
- WebSphere MQ , version 6.0
- WebSphere MQ , version 5.3
- WebSphere MQ , version 5.2
- WebSphere Process Server, version 6.0

To use this method of mapping complex products, after installing the fix pack on the administration server database computer, navigate to the directory: `<TLCM_INSTALL_DIR>/admin/db/db2`, connect to the TLMA database, and launch the script `linkmapper_crtable.sql`, as follows:

db2 -tvf linkmapper_crtable.sql

This script creates a new table in the administration server database, ADM.AGENT_CPRODUCT_MAP, which will contain an entry for each complex product component that is installed on an agent. It only needs to be run once.

The following tasks must be run on a regular basis (for example, weekly) to attempt to make the automatic associations between deployed complex product components based on the co-location rule and to process the resulting associations and use them to update the ADM.LINK table.

1. Connect to the TLMA database.

2. Navigate to the directory `<TLCM_INSTALL_DIR>/admin/db/db2` and launch the script `linkmapper_pptable.sql`, as follows:

   db2 -tvf linkmapper_pptable.sql

   This script attempts to make the associations between complex product components for each agent, using the co-location rule and adds entries to the ADM.AGENT_CPRODUCT_MAP table.
3. Ensure that the file `dblinkmapper.sh` (UNIX) or `dblinkmapper.bat` (Windows) is in the directory `<TLCM_INSTALL_DIR>/admin/db/db2` and that it has permissions set at 755.
4. Check the `dblinkmapper` file and ensure that the variables are set correctly:

   TLM_DB_JAVA_ LOG_FILE
   The path name for the trace file of the tool. Ensure that the file and path have the correct permissions.

   TLM_DB
   The name of the administration server database, normally TLMA.

   TLM_INST_PATH
   The path where Tivoli License Compliance Manager is installed.

   DB_USER, DB_PASSWORD
   The credentials required to access the database. You can leave these blank if you are running the tool as db2inst1 or any other user with access rights to the database.

   ZIP Fully qualified path name for the `db2java.zip` file that is installed on the administration server database computer.

   DB_DRIVER, DB_URL
   The name and URL of the JDBC driver. The default value works for local connections.
5. Stop the administration server.
6. Back up the administration server database.
7. Navigate to the directory `<TLCM_INSTALL_DIR>/admin/db/db2` and run `dblinkmapper.sh` or `dblinkmapper.bat`.
8. Restart the administration server.

Hardware inventory collection

Hardware inventory collection is an optional feature that enables Tivoli License Compliance Manager to collect and store hardware configuration information for hardware reconciliation processing by IBM Maximo Asset Management for IT.

You can use commands, available from the administration command line interface, to perform the following tasks:

- Enable the hardware inventory scan feature. This feature is disabled by default. See “Enabling the hardware inventory scan feature” on page 29.
- Define rules for the generation of a Globally Unique Identifier (GUID) for each monitored computer. Use of a GUID simplifies the reconciliation of hardware information with the system to which it relates. See “Defining the rule for generating GUIDs” on page 29.
- Configure hardware inventory scanning. This includes scheduling scans and determining whether to collect information in virtualized environments. See “Configuring the hardware inventory scan” on page 30.

- Check the current values for the hardware inventory scan configuration. See "Checking the configuration" on page 32.
- Check the status of hardware inventory scans. See "Checking the scan status" on page 32.

Notes:

1. You cannot activate hardware inventory collection if you are using either Tivoli License Compliance Manager for IBM software or IBM Tivoli Configuration Manager License Manager extension.
2. Hardware inventory scanning is not implemented on i5/OS® platforms or on Linux® platforms running on zSeries®.

An initial hardware scan is automatically performed as soon as the agent is deployed. Changes that imply a migration of the agent, for example changing disks or upgrading the CPU do not trigger an automatic scan. To obtain consistently updated information about changes in hardware configuration, you must schedule scans to run on a regular basis or request a one-off scan after you have made changes.

Enabling the hardware inventory scan feature

Following installation of the fix pack, the hardware inventory scan feature is disabled by default. You can enable the feature using the **hwscanenable** command.

Command syntax

hwscanenable {-d | -e}

To enable the feature, issue the command:

hwscanenable -e

Defining the rule for generating GUIDs

You can configure Tivoli License Compliance Manager to generate a GUID for each monitored computer using a specified rule. The following options are available:

None Do not generate a GUID

CSProduct
Generate a GUID by concatenating the values for manufacturer, model, and serial number.

PrimaryMACAddress
Generate a GUID using the Media Access Control (MAC) address of the first network card discovered on the computer.

Note: Neither method is able to generate unique GUIDs for systems running inside virtual containers or LPARs.

Use the **hwscanguidrule** command to set rule for generating a GUID. The default value is **None**.

Command syntax

hwscanguidrule {None | CSProduct | PrimaryMACAddress}

For example, to set the rule to CSProduct, , issue the command:

hwscanguidrule CSProduct

Note: Any change you make to the rule takes effect the next time a hardware scan is performed on each agent.

Configuring the hardware inventory scan

Configuration of the hardware inventory scan includes specifying whether agents running in virtualized environments are to perform the scan and defining the scan schedule. Both types of configuration can be applied to all the agents for a specified organization or limited to those within a specified division of the organization.

Command syntax

The command for configuring hardware inventory scans is **hwscanconf**. It has the following syntax:

hwscanconf {-organization | -division} {-n <name> | -i <*ID*>} [-S <*start_time*> -N {1 | 7 | 30} -R <*repetitions*>] [{-e | -d}]

where:

-organization
: specifies that the type of target for which the configuration is being defined is an organization. The organization name is specified using the **-n** parameter.

-division
: specifies that the type of target for which the configuration is being defined is a division. The division name is specified using the **-n** parameter. If you have multiple organizations that include divisions with the same name, you can uniquely identify the division by specifying the ID, using the **-i** parameter.

-n *name*
: specifies the name of the organization or division.

-i *ID*
: specifies the unique ID of the division.

-S *start_time*
: Specifies the date and time when the first or only occurrence of a hardware scan is start. The format *start_time* must be YYYY-MM-DD-hh.mm.ss. When the command is used to schedule a hardware scan, this parameter is required.

-N
: Specifies the number of days included in the repeating period for a regular scan. Possible values are 1, 7, and 30. This parameter can only be specified if **-S** and **-R** are also specified.

-R *repetitions*
: specifies the number repeating period that separate consecutive occurrences of the scan. The length of each repeating period is defined by the **-N** parameter. This parameter can only be specified if **-S** and **-N** are also specified.

-e
: specifies that the hardware scan is enabled in virtualized environments.

-d
: specifies that the hardware scan is disabled in virtualized environments.

Specifying the rules for running the hardware scan in virtualized environments

Once the hardware inventory scan feature is enabled, the default configuration enables collection of information from both physical and virtual environments. However, when the hardware scan runs in a virtualized environment, it collects information related to the virtual machine rather than information about the physical hardware on which the virtual machine is running and this information is not required by IBM

Maximo Asset Management for IT. You can collect and store this information for your own use or you can use the configuration command to define rules that disable the scan on agents running in virtualized environments.

The following supported environments are considered to be virtualized:

- HP Integrity VM guests
- HP vPar
- VMware guests
- Microsoft® Virtual Server guests
- Solaris Containers (Zones) guests

Note: LPAR is not treated as a virtualized environment.

To specify that agents of the Sales division that are running in virtualized environments are not to perform the scan, issue the following command:

hwscanconf -division -n Sales -d

If your environment includes multiple organizations and more than one organization has a Sales division, a message is shown, listing each Sales division with the name of the organization to which it belongs and its unique division ID. Reissue the command, using the division ID, as follows:

hwscanconf -division -i <*ID*> -d

To reenable scanning of virtualized environments, issue the command replacing the **-d** parameter with **-e**.

Scheduling hardware inventory scans

Using the configuration command, you can schedule a single hardware inventory scan or set up a series of regularly repeating scans.

To set up the schedule, you must specify the date and time on which the first or only scan is to be performed. The scheduled time is the agent system time at which the scan is to be completed.

When setting up a repeating schedule you must also define the frequency of scans.

To schedule a single occurrence of a hardware inventory scan for agents in the Sales division, at the agent system time of midnight on 1st June 2007, issue the following command:

hwscanconf -division -n Sales -S 2007-06-01-00.00.00

Scans can be scheduled to repeat at regular intervals that are multiples of 1, 7, or 30 days. To schedule a series of hardware inventory scans for agents in the Sales division, to start at agent system time of midnight on 1st June 2007 and repeat every 4 weeks, issue the following command:

hwscanconf - division -n Sales -S 2007-06-01-00.00.00 -N 7 -R 4

If there is more than one Sales division, the command fails. Reissue the command, specifying **-i** parameter and the division ID.

Checking the configuration

You can use the **hwscanshow** report to request a report of the rules and schedules you created using the **hwscanconf** command. The report can include an entire organization or can be limited to a specified division. It can be produced in either XML or CSV format.

Command syntax

The **hwscanshow** command has the following syntax:

hwscanshow <*format*> <*filename*> **{-organization | -division} {-n** <*name*> **| -i** <*ID*>**}**

where:

format
: is the format in which the report is to be produced. Possible values are CSV and XML.

filename
: is the name of the output file to be produced.

-organization
: specifies that the type of target for which the configuration is being shown is an organization. The organization name is specified using the **-n** parameter.

-division
: specifies that the type of target for which the configuration is being shown is a division. The division name is specified using the **-n** parameter. If you have multiple organizations that include divisions with the same name, you can uniquely identify the division by specifying the ID, using the **-i** parameter.

-n *name*
: specifies the name of the organization or division.

-i *ID*
: specifies the unique ID of the division.

Checking the hardware inventory scan configuration for a specified organization

To produce a report, in XML format, of the rules and schedules set up for the SafeBank organization, issue the following command:

hwscanshow XML invrules.xml -organization -n SafeBank

Checking the scan status

You can use the **hwscanstatus** command to request a report detailing when the last scan was scheduled and when the scan was actually last performed for each agent. The report can include agents for an entire organization or can be limited to a specified division. It can be produced in either XML or CSV format. You can request that the report include only those agents where the last scheduled scan was not performed, that is, agents where the last scan time is earlier than the last scheduled scan time.

Command syntax

The **hwscanstatus** command has the following syntax:

hwscanstatus <*format*> <*filename*> **[-p] {-organization | -division} {-n** <*name*> **| -i** <*ID*>**}**

where:

format
: is the format in which the report is to be produced. Possible values are CSV and XML.

filename
: is the name of the output file to be produced.

-p
: If specified, the report includes only agents where the last scheduled scan was not performed.

-organization
: specifies that the type of target for which the status is being shown is an organization. The organization name is specified using the **-n** parameter.

-division
: specifies that the type of target for which the status is being shown is a division. The division name is specified using the **-n** parameter. If you have multiple organizations that include divisions with the same name, you can uniquely identify the division by specifying the ID, using the **-i** parameter.

-n *name*
: specifies the name of the organization or division.

-i *ID*
: specifies the unique ID of the division.

Checking the hardware inventory scan status for a specified organization

To produce a report, in CSV format, of the agents in the SafeBank organization that did not successfully perform the last scheduled hardware inventory scan, issue the following command:

```
hwscanstatus CSV hwinvstatus.csv -p -organization -n SafeBank
```

Database changes

The hardware inventory scan feature adds the following tables to the administration server database:

- “The TLMHW.COMPONENT table” on page 34: Stores common system information about monitored systems.
- “The TLMHW.COMPUTER table” on page 35: Stores common system information about monitored systems.
- “The TLMHW.COMPUTER_SYS_MEM table” on page 36: Stores information about installed memory on monitored systems.
- “The TLMHW.HDISK table” on page 37: Stores information about storage devices on monitored systems.
- “The TLMHW.INST_MODEM table” on page 37: Stores information about modems installed on monitored systems.
- “The TLMHW.INST_MOUSE table” on page 37: Stores information about pointing devices on monitored systems.
- “The TLMHW.INST_PARTITION table” on page 38: Stores information about disk partitions on monitored systems.
- “The TLMHW.INST_PRINTER table” on page 38: Stores information about printers attached to monitored systems.
- “The TLMHW.INST_PROCESSOR table” on page 39: Stores information about processors on monitored systems.
- “The TLMHW.INST_SMBIOS_DATA table” on page 39: Stores general SMBIOS information for monitored systems.
- “The TLMHW.INST_USB_DEV table” on page 40: Stores information about USB devices on monitored systems.
- “The TLMHW.INST_VID_CARD table” on page 40: Stores information about video cards on monitored systems.

- "The TLMHW.IP_ADDR table" on page 40: Stores information about IP addresses on monitored systems.
- "The TLMHW.IPX_ADDR table" on page 41: Stores information about IPX addresses on monitored systems.
- "The TLMHW.MEM_MODULES table" on page 41: Stores information about all memory modules installed in a monitored system.
- "The TLMHW.MODEM table" on page 42: Stores the details for types and models of modem.
- "The TLMHW.MONITOR table" on page 42. Stores information about all monitors connected to monitored systems.
- "The TLMHW.MOUSE table" on page 43. Stores the details for types and models of pointing device.
- "The TLMHW.NET_ADAPTER table" on page 43: Stores information about network adapters on monitored systems.
- "The TLMHW.PCI_DEV table" on page 44: Stores information about peripheral component interconnect (PCI) devices on monitored systems.
- "The TLMHW.PC_SYS_PARAMS table" on page 44: Stores BIOS information for monitored PCs.
- "The TLMHW.PRINTER table" on page 45: Stores the details of types and models of printer.
- "The TLMHW.PROCESSOR table" on page 45: Stores the details of types and models of processor.
- "The TLMHW.SMBIOS_SYS_DATA table" on page 47: Stores information SMBIOS configurations.
- "The TLMHW.STORAGE_DEV table" on page 47: Stores information about storage devices on monitored systems.
- "The TLMHW.UNIX_SYS_PARAMS table" on page 47: Stores system parameter information for monitored UNIX systems.
- "The TLMHW.USB_DEV table" on page 48: Stores information about types of USB device.
- "The TLMHW.VID_CARD table" on page 48: Stores information about types of video card.

Additions have been made to several tables in the administration server database to support the hardware inventory scan feature. See "Changed tables" on page 49.

The TLMHW.COMPONENT table: Stores information about hardware components.

Columns	Description	Type	Null
ID	The component ID	bigint	no
MODEL	The component model.	varchar (128)	yes
SERIAL_NUMBER	The serial number of the component.	varchar (64)	yes
MANUFACTURER	The manufacturer of the component	varchar (64)	yes
VERSION	The component version.	varchar (64)	yes
TYPE	The component type.	varchar (64)	yes
MODEL_CLASS	The model class of the component	varchar (64)	yes

Columns	Description	Type	Null
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.COMPUTER table: Stores common information about a computer system. One entry exists for each system scanned.

Table 4. The TLMHW.COMPUTER table

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
COMPUTER_SCANTIME	The date and time of the most recent hardware scan in Greenwich Mean Time.	timestamp	yes
COMPUTER_MODEL	The model of the system.	varchar (128)	yes
COMPUTER_BOOT_TIME	The date and time that the system was last restarted.	timestamp	yes
COMUTER_ALIAS	The host name of the system.	varchar (255)	yes
SYS_SER_NUM	The serial number of the system.	varchar (64)	yes
OS_NAME	The specific operating system that is installed, for example Windows Server 2003 Standard Edition.	varchar (128)	yes
OS_TYPE	The type of operating system that is installed, for example Windows Server 2003.	varchar (32)	yes
OS_MAJOR_VERS	The major version of the operating system.	integer	yes
OS_MINOR_VERS	The minor version of the operating system.	integer	yes
OS_SUB_VERS	The sub-version of the operating system.	varchar (32)	yes
OS_INST_DATE	The date when the operating system was installed.	varchar (32)	yes
REGISTERED_OWNER	The name of the owner of the system	varchar (255)	yes
REGISTERED_ORG	The name of the organization to which the system belongs.	varchar (255)	yes
KEYBOARD_TYPE	A description of the type of keyboard assigned to the system.	varchar (255)	yes
FUNCTION_KEYS	The number of function keys on the keyboard.	integer	yes
TZ_LOCALE	The locale of the time zone in which the system is located.	varchar (32)	yes
TZ_NAME	The name of the time zone in which the system is located.	varchar (64)	yes
TZ_DAYLIGHT_NAME	The name of the daylight saving time zone in which the system is located.	varchar (64)	yes
ON_SAVINGS_TIME	Indicates whether the system is on daylight saving time.	char (1)	yes
TZ_SECONDS	The number of seconds difference between the system time zone and Greenwich Mean Time.	integer	yes

Table 4. The TLMHW.COMPUTER table (continued)

Columns	Description	Type	Null
TIME_DIRECTION	Indicates whether the time zone is early or later than Greenwich Mean Time.	char (1)	yes
OS_ARCH	The operating system architecture for the system.	varchar (24)	yes
OS_KERNEL_MODE	The operating system kernel mode	varchar (32)	yes
OS_LANG_VERS	The operating system language version identifies the default language for the system.	varchar (64)	yes
OS_LCID	The operating system locale identifier	varchar (64)	yes
CURRENT_LCID	The current locale identifier	varchar (64)	yes
VM_CHAIN_HASH	The hash of the virtual layer that hosts the agent.	char (24)	yes
VIRTUAL_COMP_TYPE	Indicates whether the monitored system is virtualized. Possible values are 0 (not virtualized) or 1 (virtualized).	smallint	no
COMPUTER_GUID	The Globally Unique Identifier (GUID) of the system.	char (32)	yes
COMPUTER_GUID_TYPE	Indicates the rule for generation of the GUID that was in use when the current information was collected. Possible values are: **1** Concatenate the values for manufacturer, model, and serial number. **3** Use the Media Access Control (MAC) address of the first network card discovered on the system.	integer	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.COMPUTER_SYS_MEM table: Stores information about the physical and virtual memory installed on a system. One entry exists for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
PHYSICAL_TOTAL_KB	The total physical memory in KB.	bigint	yes
PHYSICAL_FREE_KB	The amount of free physical memory in KB	bigint	yes
TOTAL_PAGES	The total number of physical memory pages	bigint	yes
FREE_PAGES	The number of free physical memory pages.	bigint	yes
PAGE_SIZE	The size of a page.	bigint	yes
VIRT_TOTAL_KB	The total amount of virtual memory in KB	bigint	yes
VIRT_FREE_KB	The amount of free virtual memory in KB.	bigint	yes
CREATE_TIME	The date and time when the entry was added to the database.	timestamp	no

Columns	Description	Type	Null
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.HDISK table: Stores information about storage devices, for example hard disks and CD ROM on monitored systems. One record exists for each hard disk discovered on each system scanned.

Columns	Description	Type	Null
HDISK_ID	Unique identifier of the storage device.	varchar (32)	no
HDISK_CYLINDERS	The number of cylinders in the device.	integer	yes
HDISK_SECTORS	The number of sectors in the device.	integer	yes
HDISK_HEADS	The number of heads in the device.	integer	yes
HDISK_SIZE_MB	The size of the device in MB.	bigint	yes
CREATE_TIME	The date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.INST_MODEM table: Stores information about the type of modem installed and the settings that relate to it. One record exists for each modem discovered on each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
MODEM_ID	The modem identifier.	varchar (32)	no
INST_MODEM_ID	Identifies the specific modem installed on the system.	integer	no
PORT	The port used by the modem.	varchar (16)	yes
PORT_SPEED	The speed of the port that the modem is using.	integer	yes
PORT_SETTINGS	The port settings of the modem.	varchar (16)	yes
USER_INIT	The user-specified initialization string for the modem.	varchar (128)	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.INST_MOUSE table: Stores information about the type of pointing device installed and the settings that relate to it. One record exists for each pointing device discovered on each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
MOUSE_ID	The mouse identifier.	varchar (32)	no

Columns	Description	Type	Null
INST_MOUSE_ID	Identifies the specific mouse installed on the system.	integer	no
CREATE_TIME	The date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.INST_PARTITION table: Stores information about a disk partition on a drive on the system. One record exists for each partition for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
FS_ACCESS_POINT	The location where the partition is mounted.	varchar (254)	no
DEV_NAME	The device name.	varchar (64)	yes
PARTITION_TYPE	Identifies the type of partition.	varchar (32)	yes
MEDIA_TYPE	The media type of the partition.	varchar (32)	yes
PHYSICAL_SIZE_KB	The physical size of the partition in KB	bigint	yes
FS_TYPE	The type of file system in the partition.	varchar (64)	yes
FS_MOUNT_POINT	The mount point of the file system.	varchar (254)	yes
FS_TOTAL_SIZE_KB	The total size of the file system in KB	bigint	yes
FS_FREE_SIZE_KB	The amount of free space in the file system in KB	bigint	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.INST_PRINTER table: Stores information about the type of printer installed, the driver software, and port settings. One record exists for each printer for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
PRINTER_ID	Identifies the type of printer.	varchar (32)	no
INST_PRINTER_ID	Identifies the specific printer.	integer	no
PRINTER_NAME	The name of the printer.	varchar (254)	yes
PRINTER_LOCATION	The physical location of the printer.	varchar (254)	yes
PRINTER_IS_LOCAL	Specifies whether or not the printer is local.	char (1)	yes
DRV_NAME	The printer driver name.	varchar (254)	yes
DRV_VERS	The printer driver version.	varchar (254)	yes
PORT_NAME	The name of the printer port.	varchar (254)	yes

Columns	Description	Type	Null
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.INST_PROCESSOR table: Stores information about the type of processor or processors installed. One record exists for each processor for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
PROCESSOR_NUM	The processor number	integer	no
PROCESSOR_ID	Identifies the type of processor	varchar (32)	no
SER_NUM	The processor serial number.	varchar (64)	yes
PROCESSOR_BOARD	The number of processor boards.	integer	yes
PROCESSOR_MODULE	The number of processor modules.	integer	yes
IS_ENABLED	Indicates whether the processor is enabled.	char (1)	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.INST_SMBIOS_DATA table: Stores general SMBIOS information for monitored systems. One entry exists for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
SMBIOS_ID	The SMBIOS ID	varchar (32)	no
BIOS_DATE	The date the system BIOS was created.	varchar (16)	yes
SYS_SER_NUM	The serial number of the system	varchar (64)	yes
SYS_UUID	The universal unique ID of the system	varchar (36)	yes
BOARD_SER_NUM	The serial number of the system board.	varchar (64)	yes
CASE_SER_NUM	The serial number of the case for the system.	varchar (64)	yes
CASE_ASSET_TAG	The asset tag number of the case for the system.	varchar (64)	yes
POWERON_PASSWORD	Determines whether the power on password is set.	varchar (64)	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.INST_USB_DEV table: Stores information about the type of USB devices installed. One record exists for each USB device for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
USB_ID	Identifies the type of USB device.	varchar (32)	no
INST_USB_ID	Identifies the specific USB device.	integer	no
HOST_CNTRL	The host controller for the USB device.	varchar (64)	yes
DEV_ADDR	The device address for the USB device.	varchar (16)	yes
SER_NUM	The serial number for the USB device.	varchar (64)	yes
PORT_NUM	The port number used by the USB device.	varchar (16)	yes
PARENT_ADDR	The parent address used by the USB device.	varchar (16)	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.INST_VID_CARD table: Stores information about the type of video card installed and the operating system settings. One record exists for each video card for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
VID_CARD_ID	Identifies the type of video card.	varchar (32)	no
INST_VID_CARD_ID	Identifies the specific video card.	integer	no
VID_HORIZNTL_RES	The horizontal resolution setting of the installed video card.	integer	yes
VID_VERTICAL_RES	The vertical resolution setting of the installed video card.	integer	yes
VID_COLORS	The color setting of the installed video card.	integer	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.IP_ADDR table: Stores information about the IP address for the monitored system. One record exists for each IP address for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
IP_ADDR	The IP address of the system.	varchar (40)	no
INST_IPADR_ID	Identifies the specific IP address	integer	no
IP_HOSTNAME	The IP host name for the system.	varchar (64)	yes

Columns	Description	Type	Null
IP_DOMAIN	The IP domain name for the system.	varchar (64)	yes
IP_SUBNET	The IP subnet for the system.	varchar (40)	yes
IP_GATEWAY	The IP gateway name for the system.	varchar (254)	yes
IP_PRIMARY_DNS	The primary domain name service (DNS) for the system.	varchar (40)	yes
IP_SECONDARY_DNS	The secondary DNS for the system.	varchar (40)	yes
IS_DHCP	Indicates whether the Dynamic Host Configuration Protocol (DHCP) is in use.	char (1)	yes
PERM_MAC_ADDRESS	The permanent MAC address of the system.	varchar (64)	yes
IPV6_ADDRESS	The IPV6 address of the system	varchar (64)	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.IPX_ADDR table: Stores information about the IPX address for the monitored system. One record exists for each IPX address for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
IPX_ADDR	The IPX address of the system.	varchar (40)	no
NET_NUM	The network number of the system.	varchar (40)	yes
NODE_ADDR	The node address of the system.	varchar (40)	yes
LINK_SPEED	The link speed of the system.	integer	yes
MAX_PACKET_SIZE	The maximum package size that the system can handle.	integer	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.MEM_MODULES table: Stores the details of memory modules installed. One entry exists for each memory module for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
INST_MEM_ID	The index number for the memory module.	integer	no
MODULE_SIZE_MB	The size of the installed memory module in MB.	integer	yes
MAX_MODULE_SIZE	The maximum supported memory module size in MB.	integer	yes

Columns	Description	Type	Null
SOCKET_NAME	The name of the socket in which the memory module is installed	varchar (24)	yes
PACKAGING	The physical packaging of the memory, such as single in-line memory module (SIMM) or dual in-line memory module (DIMM).	varchar (16)	yes
MEM_TYPE	The type of memory installed.	varchar (48)	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.MODEM table: Stores the details for one particular type and model of modem. It contains one record for each unique modem in use in the monitored environment.

Columns	Description	Type	Null
MODEM_ID	The modem identifier.	varchar (32)	no
MODEM_DESC	The description of the modem.	varchar (64)	no
MANUFACTURER	The manufacturer of the modem.	varchar (64)	yes
PROVIDER_NAME	The maker of the software driver for the modem.	varchar (32)	yes
MODEM_TYPE	The type of modem.	varchar (32)	yes
INF_FILE	The name of the description file for the modem driver.	varchar (32)	yes
INF_SECTION	The details of the driver file for the modem.	varchar (32)	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.MONITOR table: Stores the details of monitors connected to systems in the monitored environment. One entry exists for each monitor for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
INST_MONITOR_ID	Identifier of the installed monitor.	integer	no
MANUFACT_CODE_NAME	Identifies the manufacturer of the monitor.	varchar (64)	yes
NAME	The name of the manufacturer	varchar (64)	yes
WEEK_MANUFACTURED	The week in which the monitor was manufactured.	integer	yes
YEAR_MANUFACTURED	The year in which the monitor was manufactured.	integer	yes
SERIAL_NUMBER	The serial number of the monitor.	varchar (64)	yes
SIZE_INCHES	The size of the monitor in inches	integer	yes

Columns	Description	Type	Null
MIN_V_SCAN_FREQ	The minimum vertical scan frequency of the monitor.	integer	yes
MAX_V_SCAN_FREQ	The maximum vertical scan frequency of the monitor.	integer	yes
MIN_H_SCAN_FREQ	The minimum horizontal scan frequency of the monitor.	integer	yes
MAX_H_SCAN_FREQ	The maximum horizontal scan frequency of the monitor.	integer	yes
PIXEL_CLOCK_MHZ	The rate at which pixels are drawn on the screen.	integer	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.MOUSE table: Stores the details for one particular type and model of pointing device. It contains one record for each unique pointing device in use in the monitored environment.

Columns	Description	Type	Null
MOUSE_ID	The mouse identifier.	varchar (32)	no
MOUSE_MODEL	The model of mouse.	varchar (254)	yes
MOUSE_TYPE	The type of pointing device for example mouse, trackball.	varchar (32)	yes
BUTTONS	The number of buttons on the mouse.	integer	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.NET_ADAPTER table: Stores information about the network adapter installed on a system. One record exists for each network adapter for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
INDEX	The net adapter index.	integer	no
PERM_MAC_ADDR	The permanent Media Access Control (MAC) address of the system.	varchar (64)	yes
CURRENT_ADDR	The current network address of the system.	varchar (64)	yes
ADAPTER_TYPE	The network adapter installed on the system.	varchar (32)	yes
ADAPTER_MODEL	The model of the network adapter	varchar (254)	yes
MANUFACTURER	The manufacturer of the network adapter.	varchar (128)	yes
INST_DATE	Network adapter installation date.	varchar (32)	yes

Columns	Description	Type	Null
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.PCI_DEV table: Stores information about a PCI device installed in or connected to a system. One record exists for each PCI device for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
INST_PCI_ID	Identifies the installed peripheral component interconnect (PCI) device.	integer	no
PCI_DEV_NAME	The name of the PCI device.	varchar (180)	no
PCI_REVISION	The revision of the PCI device.	char (8)	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.PC_SYS_PARAMS table: Stores BIOS and other system information for a PC. One entry exists for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
USER_NAME	The Windows user name for the system.	varchar (254)	yes
DOMAIN_NAME	The Windows domain name for the system.	varchar (128)	yes
WORKGROUP_NAME	The Windows workgroup name for the system.	varchar (128)	yes
BIOS_ID	Identifies the BIOS that is in use on the system.	varchar (128)	yes
BIOS_ID_BYTES	Hexadecimal values from the BIOS.	varchar (16)	yes
BIOS_DATE	The revision date of the BIOS.	varchar (32)	yes
BIOS_STRING	The BIOS string.	varchar (128)	yes
BIOS_MANUFACTURER	Identifies the manufacturer of the BIOS.	varchar (64)	yes
MANUFACTURER_ID	Identifies the manufacturer of the PC.	varchar (64)	yes
BIOS_MODEL	The BIOS model.	varchar (64)	yes
BIOS_SER_NUM	The BIOS serial number.	varchar (64)	yes
UPTIME	The current uptime of the system. Indicates the level of reliability of the system.	bigint	yes
IE_VERS	The version of Microsoft Internet Explorer that is installed.	varchar (64)	yes

Columns	Description	Type	Null
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.PRINTER table: Stores the details for one particular type and model of printer. It contains only one record for each unique printer in use in the monitored environment.

Columns	Description	Type	Null
PRINTER_ID	Unique identifier of the printer type.	varchar (32)	no
PRINTER_MODEL	A description of the printer model.	varchar (254)	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.PROCESSOR table: Stores the details for one particular type and model of processor. It contains only one record for each unique processor in use in the monitored environment.

Columns	Description	Type	Null
PROCESSOR_ID	Unique identifier of the processor type.	varchar (32)	no
MANUFACTURER	The manufacturer of the processor.	varchar (128)	yes
PROCESSOR_MODEL	The processor model.	varchar (200)	yes
PROCESSOR_FEATURES	A bitmask within which each bit that is set identifies a feature that is available for the processor.	integer	yes
MAX_SPEED	The maximum speed of the processor in MHz.	integer	yes
CURRENT_SPEED	The current speed of the processor in MHz.	integer	yes
BUS_SPEED	The external bus speed of the processor in megahertz (MHz).	integer	yes
CPU_INTERFACE	The external central processing unit (CPU) packaging interface.	varchar (32)	yes
ECACHE_MB	The size of the processor ecache in MB	varchar (16)	yes
CPU_IMPL	The implementation (type) of the processor	varchar (16)	yes
CPU_MASK	The CPU mask for the processor.	varchar (16)	yes
CHIP_FAMILY	The chip family of the processor.	integer	yes
CHIP_MODEL	The chip model of the processor.	integer	yes
CHIP_STEPPING	The chip stepping code for the processor	integer	yes
VIRT_MODE_EXT	Indicates whether virtual mode extension is supported.	char (1)	yes
PAGE_SIZE_EXT	Indicates whether page size extension is supported.	char (1)	yes

Columns	Description	Type	Null
TIME_STAMP_COUNTER	Indicates whether time stamp counter is supported.	char (1)	yes
MODEL_SPECIFIC_REG	Indicates whether model-specific registers are supported.	char (1)	yes
PHYSICAL_ADDR_EXT	Indicates whether physical address extension is supported.	char (1)	yes
MACHINECHECK_EXCPT	Indicates whether machine-check exception is supported.	char (1)	yes
CMPXCHG8B_SUPP	Indicates whether the compare exchange 8-byte instruction is supported.	char (1)	yes
ON_CHIP_APIC	Indicates whether the integrated advanced programmable interrupt controller (APIC) is supported.	char (1)	yes
MEM_TYPE_RANGE_REG	Indicates whether memory type range registers are supported.	char (1)	yes
PAGE_GLOBAL_ENABLE	Indicates whether the page global extension is enabled.	char (1)	yes
MACHINECHECK_ARCH	Indicates whether the machine-check architecture is supported.	char (1)	yes
COND_MOVE_SUPP	Indicates whether the conditional move instruction is supported.	char (1)	yes
MMX_TECHNOLOGY	Indicates the Intel® MMX features that are supported.	char (1)	yes
ON_CHIP_FPU	Indicates whether the floating processor unit is supported.	char (1)	yes
DEBUG_EXT_PRESENT	Indicates whether the debugging extension is supported.	char (1)	yes
FAST_SYS_CALL	Indicates whether fast system call is supported.	char (1)	yes
PAGE_ATTR_TABLE	Indicates whether the page attribute table extension is supported.	char (1)	yes
PAGE_SIZE_EXT36	Indicates whether the 36-bit page size extension is supported.	char (1)	yes
SER_NUM_ENABLED	Indicates whether the processor serial number is enabled.	char (1)	yes
FAST_FLOAT_SAVE	Indicates whether fast floating point save and restore instructions are supported.	char (1)	yes
SIMD_EXT_SUPP	Indicates whether the streaming single instruction, multiple data (SIMD) instruction set is supported.	char (1)	yes
NOW_3_D_ARCH	Indicates whether the AMD 3DNow! instructions are supported.	char (1)	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.SMBIOS_SYS_DATA table: Stores general SMBIOS details. It contains only one record for each unique SMBIOS configuration that is in use in the monitored environment.

Columns	Description	Type	Null
SMBIOS_ID	SMBIOS ID.	varchar (32)	no
BIOS_VENDOR	The manufacturer of the system BIOS.	varchar (64)	yes
BIOS_VERS	The version number of the system BIOS.	varchar (128)	yes
BIOS_SIZE	The size of the system BIOS.	char (8)	yes
SYS_MANUFACTURER	The manufacturer of the system.	varchar (64)	yes
SYS_PRODUCT_NAME	The product name of the system.	varchar (64)	yes
SYS_VERS	The version number of the system.	varchar (64)	yes
BOARD_MANUFACTURER	The manufacturer of the system board.	varchar (64)	yes
BOARD_PRODUCT	The product name of the system board.	varchar (64)	yes
BOARD_VERS	The version number of the system board.	varchar (64)	yes
CASE_MANUFACTURER	The manufacturer of the case for the system.	varchar (64)	yes
CASE_TYPE	The type of case for the system.	varchar (64)	yes
CASE_VERSION	The version number of the case for the system.	varchar (64)	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.STORAGE_DEV table: Stores the details for a particular storage device installed on a system. One record exists for each storage device for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
INST_STORAGE_ID	The index number for the installed storage device	integer	no
STORAGE_TYPE	The type of storage device.	varchar (64)	yes
MANUFACTURER	The manufacturer of the storage device.	varchar (128)	yes
MODEL	The model of storage device.	varchar (254)	yes
SER_NUM	The serial number of the storage device	varchar (64)	yes
HDISK_ID	The hard disk ID	varchar (32)	no
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.UNIX_SYS_PARAMS table: Stores UNIX system parameters for a system. One entry exists for each system scanned.

Columns	Description	Type	Null
COMPUTER_SYS_ID	Unique identifier of the system.	bigint	no
BOOT_TIME	The date and time that the system was last restarted.	timestamp	yes
UPTIME	The current uptime of the system. Indicates the level of reliability of the system.	integer	yes
RUN_LEVEL	The run level of the system.	integer	yes
HOST_NAME	The host name of the system.	varchar (254)	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.USB_DEV table: Stores the details for one particular type of USB device. It contains only one record for each unique USB device in use in the monitored environment.

Columns	Description	Type	Null
USB_ID	The USB ID	varchar (32)	no
USB_VERS	The USB version of the device.	varchar (32)	yes
DEV_CLASS	The device class of the USB device.	varchar (32)	yes
DEV_SUBCLASS	The device subclass for the USB device.	integer	yes
VENDOR_ID	The vendor ID of the manufacturer of the USB device.	integer	yes
PRODUCT_ID	The product ID for the USB device.	integer	yes
MANUFACTURER	The manufacturer for the USB device.	varchar (128)	yes
PRODUCT	The type of product the USB device is.	varchar (254)	yes
NUM_OF_PORTS	The number of USB ports present on the USB device.	integer	yes
DEV_IS_HUB	Indicates whether the USB device is a hub.	char (6)	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

The TLMHW.VID_CARD table: Stores the details for one particular type of video card. It contains only one record for each unique video card in use in the monitored environment.

Columns	Description	Type	Null
VID_CARD_ID	The video card ID	varchar (32)	no
VID_CARD_MODEL	The video card model.	varchar (254)	yes
VID_CARD_BIOS	The BIOS information for the video card.	varchar (254)	yes

Columns	Description	Type	Null
VID_DAC_TYPE	The integrated digital-to-analog converter (DAC) for the video card.	varchar (254)	yes
VID_MEM	The amount of memory for the video card.	integer	yes
VID_BIOS_RELDATE	The BIOS release date	varchar (32)	yes
VID_CHIP_TYPE	The chip type for the video card.	varchar (254)	yes
CREATE_TIME	Date and time when the entry was added to the database.	timestamp	no
UPDATE_TIME	Date and time when the entry was last updated.	timestamp	no

Changed tables: Additions have been made to the following tables to support the hardware inventory scan feature:

- ADM.AGENT. See Table 5.
- ADM.CONTROL. See Table 6.

Table 5. Additions to the ADM.AGENT table.

Columns	Description	Type	Null
HWSCAN_TIME	The date and time of the last hardware scan.	timestamp	yes

Table 6. Additions to the ADM.CONTROL table.

Columns	Description	Type	Null
HARDWARE_SCAN _ENABLED	Indicates whether the hardware inventory scan feature is enabled.	char (5)	yes
COMPUTER_GUID_RULE	Indicates the rule used to calculate the computer GUID. Possible values are: **1** Concatenate the values for manufacturer, model, and serial number. **3** Use the Media Access Control (MAC) address of the first network card discovered on the system.	smallint	yes

Note: If you installed fix pack 2.3.0-TIV-TLCM-FP0001, additional fields will have been created in the ADM.DIVISION and ADM.DIVISION_DELETED tables, to support the hardware inventory scan feature. The introduction of the Agent Configuration feature and the database changes that it includes in fix pack 2.3.0-TIV-TLCM-FP0002 make these fields redundant and they are removed from the tables.

Agent installation size check

For all agent deployment methods, a space check is now made to ensure that the installation will not start and then fail because of lack of sufficient space in the agent installation directory. Before starting the installation process, the following space requirement in the agent installation directory must be satisfied:

Windows
53 MB

Linux 70 MB

Solaris and AIX
85 MB

HP-UX
95 MB

Note: These figures include the space requirement for GSKit, which is now installed in the agent installation path.

If the space available is insufficient, the installation fails with return code -17.

Changes to GSKit installation

This feature introduces a level of flexibility to the requirement to install the operating system patches that are required by GSKit security software.

After the installation of the fix pack, agent installation no longer fails if patches required by GSKit are not installed on the target computer. If the installation parameters specify that the agent is to use SSL when communicating with the runtime server (medium and maximum security levels), the agent installs successfully but is not started. A warning message is issued to alert you to the need to the install the missing patches or change to the minimum security level.

Table 7 lists the maintenance levels, service packs, patches, or compatibility packs, specified as agent prerequisites for Tivoli License Compliance Manager, version 2.3, that with fix pack 2 become requirements only when SSL communication is in use.

Table 7. Operating system patches required for GSKit

Platform		
Operating system	**Version**	**Levels, service packs, patches, compatibility packs**
IBM AIX	5.3 (32-bit)	xlC.aix50.rte.6.0.0.3 or later
	5.3 (64-bit)	
	5.2 (32-bit)	xlC.aix50.rte.6.0.0.3 or later
	5.2 (64-bit)	
HP-UX	11i v2 on PA-RISC 2.0 32-bit	Patches PHSS_26946, PHSS_33033,
	11i v2 on PA-RISC 2.0 64-bit (in 32-bit compatibility mode)	
Red Hat Enterprise Linux	ES 4.0 for Intel x86	Compatibility packs: compat-gcc-32-c++-3.2.3-46.1.i386.rpm compat-gcc-32-3.2.3-46.1.i386.rpm compat-libstdc++-33-3.2.3-46.1.i386.rpm
Sun Solaris	9 Operating System for SPARC platforms (32-bit)	111711-08
	9 Operating System for SPARC platforms (64-bit)	
	8 Operating System for SPARC platforms (32-bit)	108434-14, 108528-29, 108991, 108993-31, 111023-03, 111317-05, 111327-05, 113648-03, 115827-01, and 116602-01

The GSKit software is installed within the agent installation path and cannot be shared by other applications that are installed on the target computer.

Changes to the process for upgrading from version 2.1

This feature introduces a change to the process of moving from Tivoli License Compliance Manager, version 2.1 to version 2.3 described in *IBM Tivoli License Compliance Manager: Release Notes*. The process introduced at version 2.3 general availability supports the retention and migration of the data held in the version 2.1 administration server database, but not the data held in the runtime server database. As a result, information about existing agents is not available to the version 2.3 runtime server when it is installed. All agents are considered as new agents when they plug in. A problem arises with the version 2.1 agents. The version 2.3 runtime server provides limited support for existing version 2.1 agents, including self-update, but does not accept the addition of new agents at this version.

The solution provided with this fix pack is to provide a script to populate the runtime server database with information from the administration server database that will enable existing agents to be recognized by the runtime server.

To implement this solution, do the following;

1. Perform the procedure of administration server migration and installation of the version 2.3 servers and databases that is described in the "Installing and upgrading the product" section of *IBM Tivoli License Compliance Manager: Release Notes*.
2. Apply fix pack 2 to each runtime server database.
3. On each runtime server database computer, do the following:
   a. Catalog the administration server database so that it can be accessed when the script runs.

      Note: This step is not required if you have a single runtime server database installed on the same computer as the administration server database.

   b. Navigate to the directory `<TLCM_INSTALL>\db\db2` and launch the `migrateAgents` script as follows:

      UNIX From a db2inst1 session, launch the script:

      migrateAgents.sh <TLMA> <TLMR> <user> <password>

      Windows
      From a command window, launch the script:

      migrateAgents.bat <TLMA> <TLMR> <user> <password>

      Where:

      TLMA
      is the name of the administration server database.

      TLMR
      is the name of the runtime server database.

      user and password
      are the credentials required to connect to the administration server database. For example, users could be tlmsrv, db2inst, or db2admin.

Agent platforms

Support is introduced for the following agent platforms on IBM Power 6 processors:

- IBM AIX, version 5.3
- Red Hat Enterprise Linux AS release 4 (Nahant Update 5) for PPC

Support for the following agent platforms is reintroduced with this fix pack:

- Sun Solaris 8 (32 and 64-bit)
- Windows 2000 Advanced Server (32-bit)
- Windows 2000 Server (32-bit)
- Windows 2000 Profession (32-bit)

Service Pack 3 is required for all the Windows platforms.

Support for partitioning technologies

The fix pack introduces support for agents in Solaris Containers non-global zones with resource pools enabled.

When an agent is deployed in a non-global zone and resource pools are enabled, the agent details that are shown on the Web UI include information about the processor set associated with the resource pool for the zone where the agent is deployed.

Notes:

1. When agents are deployed in non-global zones, an agent must also be deployed in the global zone.
2. On UltraSPARC T1 processors, the agent is not able to retrieve the correct LPAR capacity for a local zone when the resource pool is enabled.

Product fixes

This is the first fix pack issued for Tivoli License Compliance Manager, version 2.3. It fixes the following APARs:

APAR IY99988

An exception occurs when running the hwscanguidrule command on UNIX platforms when the server is installed with WebSphere Application Server, version 6.1.

The fix pack resolves the problem.

APAR IY99182

While unsuccessfully deploying agents, the following misleading message is traced: "Error writing main configuration file".

The fix pack resolves the problem from a serviceability point of view by providing additional information when the failure occurs.

APAR IY98781

Tracing is not implemented for the hardware inventory scan commands introduced in fix pack 2.3.0-TIV-TLCM-FP001.

This fix pack resolves the problem.

APAR IY98762

Following the installation of fix pack 2.3.0-TIV-TLCM-FP001, the facility to export reports to XML format does not work.

This fix pack resolves the problem.

APAR IY98744
: On Solaris 10 agents, an error associated to the zoneadm command occurs when the agent is started.

  The fix pack resolves the problem.

APAR IY97252
: Exception messages are issued by the WebSphere agent when security is enabled for the WebSphere cell.

  The fix pack resolves the problem.

APAR IY96542
: On AIX, an error is logged in the error report (errpt) during agent deployment.

  The fix pack resolves the problem.

APAR IY96527
: An exception is logged in the trace files if the runtime server is not able to download an update of the catalog from the administration server.

  When this happens, an event is generated to notify the user.

APAR IY96387
: Agent deployment on i5/OS systems fails if it is not possible to generate an agent ID from the host address.

  Following installation of the fix pack, if the agent ID cannot be generated using the host address, a random ID is generated and the deployment proceeds.

APAR IY95336
: On Linux 390 platforms, the manual agent deploy setup fails in silent mode.

  The fix pack resolves the problem.

APAR IY95183
: On Windows systems, a noise is heard when the agent checks the presence of the floppy disk.

  The fix pack resolves the problem by installing a new version of the Common Inventory Technology (CIT) component.

APAR IY94558
: The initial scan performed at agent start up takes too long.

  The problem was related to the collection of system storage information. This is no longer required and so has been removed.

APAR IY94123
: The Software Package Block (SPB) for agent configuration changes does not include the possibility to change the organization.

  The fix pack adds the Organization parameter to the SPB `change_agt_configuration_platform.spb.`

  The main objective of this change is to allow configuration errors to be corrected. If the name of the organization has been entered incorrectly, the agent is not able to plug in. You can use the software package block to distribute a correction to all affected agents.

You can also use the SPB to transfer agents that are already active from one organization to another. However, be aware that all installed software and software use information that has already been collected by the agents will no longer be available. Before distributing the SPB, use the **Manage Infrastructure > Agents** task on the Web UI to delete the agent. When the agent plugs in after the SPB has been applied, a new agent record is automatically created.

APAR IY93924
The time on the agents is not in synch with the time on the server.

The fix pack resolves the problem.

APAR IY93548
No software use is detected for WebSphere Commerce 5.6.x when the J2EE application has the same name as the application server.

The fix pack resolves the problem.

APAR IY93364
If the administration and runtime servers are started using the Windows command line and, later on, the user logs off from Windows, the servers are also stopped.

The fix pack resolves the problem by using Windows services to start/stop the Tivoli License Compliance Manager servers. By doing so, the servers will stay active if the user logs off or closes the connection.

APAR IY93328
Abend code on an i5/OS agent because of a variable overflow caused by an http connectivity problem with the runtime server.

The fix pack resolves the problem.

APAR IY93251
The agent cannot handle correctly 'Manufacturer' strings longer than 31 characters.

The fix pack resolves the problem.

Problems fixed

The following problems were found since the general availability of Tivoli License Compliance Manager, version 2.3 and fixed with this fix pack:

The generateAgentId command does not work if fix pack 3 is applied to WebSphere Application Server, version 6.1
This problem is caused because the update to the Java SDK introduced in fix pack 3 is not backward compatible. At the GA version of Tivoli License Compliance Manager, version 2.3, the **generateAgentId** command will not work after you have applied fix pack 3 to WebSphere Application server.

When this fix pack has been applied, the command works both with and without fix pack 3.

Exceptions are generated when a division that has many connected agents is deleted.
When the fix pack has been applied, it is no longer possible to delete divisions that have connected agents.

Problems and workarounds

Incorrect agent installation return code reported in the Tivoli Configuration Manager logs.

When the agent is deployed using Tivoli Configuration Manager and errors occur, incorrect agent installation return codes are written to the software package block log.

This problem is caused because the error return codes are negative integers and logging in Tivoli Configuration Manager does not handle them correctly. The correct return code is written in the file `slmrc`, which is stored in the directory on the target computer from which the agent installation was launched.

On HP-UX platforms, problems can occur accessing the Web UI after the server has been stopped and restarted.

This problem is caused by a WebSphere transport problem that affects both the administration and runtime servers. The problem is resolved in fix pack 5 for WebSphere Application Server 6.1.

If you do not want to apply the WebSphere Application Server fix pack at this time, you can overcome the problem by stopping both the Tivoli License Compliance Manager server and the HTTP server and then restarting them.

If multiple commands are launched at the same time the commands fail with unpredictable error messages.

This problem occurs when multiple server command line sessions are in use because the command line interface has a limited number of database connections that it can use before other server activities are affected.

To resolve the problem and continue working with the command line, close some of the command line sessions that you have open.

Incorrect calculation of LPAR capacity when resource pools are enabled on UltraSPARC T1 processors

Agents installed on Solaris platforms running on UltraSPARC T1 processors are not able to retrieve the correct LPAR capacity for a local zone when the resource pool is enabled.

A delay occurs when transferring information using the wtlminfoget command.

When Tivoli License Compliance Manager administration server components are installed as part of the Tivoli Configuration Manager License Extension, a delay occurs before changes made on the Tivoli License Compliance Manager Web user interface can be retrieved using the Tivoli Configuration Manager command, **wtlminfoget.**

Documentation fix history

This section provides corrections, additions and clarifications to the information documented in the product manuals:

The prerequisite version of WebSphere Application Server requires clarification

The minimum prerequisite for Tivoli License Compliance Manager servers is WebSphere Application Server version 6.0 with refresh pack 2 and fix pack 5 (6.0.2.5). Fix pack 5 has two "flavours". Ensure that the fix pack you install includes fix PK21297, which includes the update to the JDK.

Missing information about installing the agent on VMware environments (APAR IY96544 opened against Tivoli License Compliance Manager version 2.2)

Problem (found in *IBM Tivoli License Compliance Manager: Planning,* *Installation, and Configuration*): you can install the agent on VMware environments without running the CIT enabler. Agents installed in this manner will only be able to send inventory data: they will not send usage data, nor aggregated or hardware information. On agents running VMware ESX 3.0 (or later), APAR IY96549 for CIT version 2.3 must be installed. The CIT enabler in this fix pack includes the APAR.

Missing CIT required libraries for VMware environments (APAR IY95943 opened against Tivoli License Compliance Manager version 2.2)

Problem: In Chapter 1 of *IBM Tivoli License Compliance Manager: Planning, Installation, and Configuration*, the information about libraries required by CIT for VMware environments is documented in Table 4 ″Supported agent platforms″, but is missing from page 20 where the CIT enabler specifications are documented.

Solution: The table that documents the CIT enabler software on page 20 of Chapter 1 should include the following new paragraph: ″On hosts running Red Hat Linux, the enabler requires the compatibility packs documented in Table 4 on page 11.″

Instructions for configuring the WebSphere Application Server virtual host for secure communications are not clear. (APAR IY94284 opened against Tivoli License Compliance Manager version 2.2)

Problem: In Chapters 2 and 3 of *IBM Tivoli License Compliance Manager: Security Management*, the instruction that describes the configuration of a secure port for all virtual host aliases does not make clear which virtual host should be selected.

Solution: In both chapters, the instruction is changed as follows:

Ensure that the WebSphere Application Server has virtual aliases for the secure port.

Logon to the WebSphere Application Server console and do the following:

1. On the Virtual Host settings page, select the virtual host instance called **default_host**.
2. Configure a secure port for the virtual host and all its aliases. You can create a single definition that applies to all possible host aliases, setting the **Host Name** to * and **Port** to the secure port number (normally 443).
3. Generate the Web Server plug-in.
4. Propagate the Web Server plug.in.

Clarification for differences in the steps for configuring Java 2 security depending on the WebSphere Application Server configuration.

Problem: In Chapter 5 of *IBM Tivoli License Compliance Manager: Security Management*, the instructions for configuring Java 2 security do not reflect possible differences in the WebSphere Application Server configuration.

Solution: The following differences need to be stated:

- In the document the directory where Java 2 policies are stored is given as:

```
<WAS_INSTALL_DIR>\profiles\default\config\cells\
<host_cell_name>\applications\
SLM_<server_type>


_Application.ear\deployments\
SLM_<server_type>_Application\META-INF
```

The subdirectory `\default` is should be documented as a variable. The subdirectory name is the name of the default profile created when WebSphere Application server is installed. It could be "default", but it could also have other values.

- If you have installed the Tivoli License Compliance Manager server with an existing installation of WebSphere Application Server, for which Java 2 security was already configured, you need only to complete the task of replacing the profile file with the file in which you have defined the location of the Tivoli Common Directory. Steps 7, 8, and 9 are not required.

The prerequisite for space in the /tmp directory during installation of the runtime server is too low (APAR IY93190 opened against Tivoli License Compliance Manager version 2.2)

Problem: In Chapter 1 of *IBM Tivoli License Compliance Manager: Planning, Installation, and Configuration*, the table of prerequisites for servers specifies a requirement for 250 MB of free space on UNIX platforms in the `/tmp` directory. For runtime server installations, this is insufficient. In addition, the table of prerequisites for the runtime server should say that the WebSphere Application Server temporary directory (java.io.tmpdir) by default is set to the `/tmp` directory.

Solution: The requirement is to have at least 700 MB of free space on UNIX platforms.

The syntax for the backupconf and restoreconf commands is incorrect.

Problem: In *IBM Tivoli License Compliance Manager: Commands*, the syntax described for the command backupconf and restoreconf commands is incorrect. The documented syntax for the commands is:

backupconf [-d <*directory_name*>**]**
restoreconf [-d <*directory_name*>**]**

The correct syntax is:

backupconf [<*directory_name*>**]**
restoreconf [<*directory_name*>**]**

Backward compatibility

Dependencies:
Tivoli License Compliance Manager, version 2.3

Other changes as a result of this fix pack

This fix pack creates or updates a file called `<INST_DIR>\product.xml` (where `<INST_DIR>` is the directory where Tivoli License Compliance Manager is installed), which keeps track of the current Tivoli License Compliance Manager version.

Agents deployed after this fix pack has been applied to the runtime server will have a new version number of 2.3.0.20. The individual agents' software will be updated only when you specifically reinstall the agent software, or after you have used the agent self–update facility in the agent configuration file, as described in *IBM Tivoli License Compliance Manager: Planning, Installation, and Configuration*.

Installation, migration, upgrade, and configuration information

This section includes the following topics:

- "Hardware and software requirements"
- "Installing the fix pack" on page 59
- "Upgrading agents" on page 60
- "Reverting to the previous version" on page 60

Hardware and software requirements

This section includes the following topics:

- "Supported platforms"
- "System requirements"

Supported platforms

The following information supplements the supported platform information documented in *IBM Tivoli License Compliance Manager: Planning, Installation, and Configuration*:

On the following platforms, changes are required to the required maintenance levels, service packs, patches or compatibility patches

HP-UX 11i v2 running on rx7640 and rx8640.
Required patch: PHKL_35174

This patch is required when agents are deployed in partitioned environments.

HP-UX 11i v2 on PA-RISC
Required patch: PHSS_35381

Limitation to the deployment of agents on Solaris x86 platforms (global and non-global zones)
Agents cannot be deployed on hardware that supports hyper-threading functionality.

System requirements

The following are additions to hardware and software prerequisites detailed in *IBM Tivoli License Compliance Manager: Planning, Installation, and Configuration*.

The prerequisite version of WebSphere Application Server requires clarification
The minimum prerequisite for Tivoli License Compliance Manager servers is WebSphere Application Server version 6.0 with refresh pack 2 and fix pack 5 (6.0.2.5). Fix pack 5 has two "flavours". Ensure that the fix pack you install includes fix PK21297, which includes the update to the JDK.

The space in the agent installation directory required for agent installation is changed

Windows agent
38 MB

Linux agent
50 MB

Solaris, AIX and HP-UX agents
65 MB

GSKit component
30 MB

Installing the fix pack

This section contains the instructions for installing the fix pack. Once you have installed the fix pack, you cannot uninstall it automatically. For details of how to revert to the previous version see "Reverting to the previous version" on page 60.

The fix pack includes the following files:

2.3.0–TIV-TLCM-FP0002-servers-<*platform*>.zip
For each supported platform, the compressed file contains the files for installing the fix pack on computers where an administration server, a runtime server, or an administration server database or runtime server database is installed.

2.3.0–TIV-TLCM-FP0002-SPB.zip
This file contains the software package blocks for each supported agent platform, to be used when deploying agents using IBM Tivoli Configuration Manager.

2.3.0-TIV-TLCM-FP0002-agent-gateway.zip
This file contains the software packages for each supported agent platform in PKT format.

2.3.0–TIV-TLCM-FP0002-RSH-SSH-<*platform*>.zip
This contains the files needed to deploy agents on UNIX platforms using the RSH/SSH deployment wizard.

2.3.0–TIV-TLCM-FP0002-agentOS400.zip
This contains the files needed to deploy agents on i5/OS platforms.

2.3.0–TIV-TLCM-FP0002-ManualDeploy-<*platform*>.zip
This contains the files needed to install agents using a local wizard.

2.3.0-TIV-TLCM-FP0002-SPB-TOOLS.zip
This contains the SPBs for agent configuration update that have been updated for APAR IY94123.

The fix pack must be applied to all server and database components of Tivoli License Compliance Manager.

Notes:

1. Before starting the installation, back up the administration and runtime server databases and the installation directories of administration server and runtime servers.
2. During the installation of the fix on the administration server or a runtime server, the wizard stops and starts the server. If the server is running in a WebSphere Application Server secure cell, you will be asked to provide the user ID and password for the secure cell. Ensure that you have this information before you start.

To install the fix pack, complete the following steps on each computer where a component is installed:

1. Log on to the computer where one or more of the Tivoli License Compliance Manager components is installed as Administrator (Windows) or root (UNIX).
2. Unpack the servers compressed file into a temporary directory.
3. Launch the setup file for the platform on which you are installing the fix pack.
   The installation wizard starts.

Note: No license agreement panel is displayed. The fix pack is subject to the same terms and conditions under which Tivoli License Compliance Manager, version 2.3 is licensed.

4. Specify a directory where the wizard can create a backup of your server configuration and other files that are affected by the fix pack and click **Next**.

   If the wizard is unable to create the backup, the installation cannot continue. Ensure that you have the correct permissions to write to the specified directory and that there is sufficient space to create the backup.

   Note: The wizard calculates the space required to create the backup based on the initial size of a server when it is installed. Under normal circumstances, the space occupied by the server directories should not increase very much. However, if other directories, for example directories required for manual agent deployment, have been created within the server file structure, the space required for the backup could be considerably more than anticipated. Under these circumstances, it is possible that the installation could fail because of lack of space. If this occurs, a message will be displayed informing you that the installation has failed, possibly because of insufficient permissions. You must then make more space available for the installation.
5. Check the summary panel for the installation and click **Next** to confirm that you want to install the fix pack.

   If you are applying the fix pack to a server that is running in a WebSphere Application Server secure cell, you will be prompted to supply a valid user ID and password when the wizard stops and starts the runtime server.
6. When the installation is complete, click **Finish**.

Upgrading agents

To apply the fixes to agents that are already deployed, you must redeploy or update the agent. For information about how to do redeploy and agent see the section **Redeploying an agent** in *IBM Tivoli License Compliance Manager: Planning, Installation, and Configuration*.

The agent self-update procedure is changed with this fix pack. For information about updating agents to this fix pack level when they are connected to a runtime server where the fix pack has been applied, see “Configuring a periodic agent self-update” on page 11.

Agents are supported only when they are registered to a runtime server with the same, or a higher, Tivoli License Compliance Manager maintenance level.

Reverting to the previous version

There is no automatic method for uninstalling this fix pack.

You must rollback the changes on each computer where the fix pack has been installed, as follows:

1. Stop the server.
2. Manually replace the files that were changed by the installation of the fix pack with the backup copies taken during the installation.

   The backup directory specified during the installation of the fix pack contains a subdirectory `2.3.0-TIV-TLCM-FP0002` which contains a subset file structure for the Tivoli License Compliance Manager components to which the fix pack has been applied. You must copy the files manually from each subdirectory of the backup to the corresponding directory of the server installation.

3. If this is the first fix applied to the GA version of the code, delete the file `<`*`INSTALL_DIR`*`>\product.xml`.

   If previous fixes have been applied, the `product.xml` file is one of the files that you have restored from the backup directory.
4. Restart the server.

Notices

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

IBM
The IBM logo
AIX
DB2
DB2 Universal Database
Tivoli
WebSphere
zSeries

Program Number: 5724-D33