

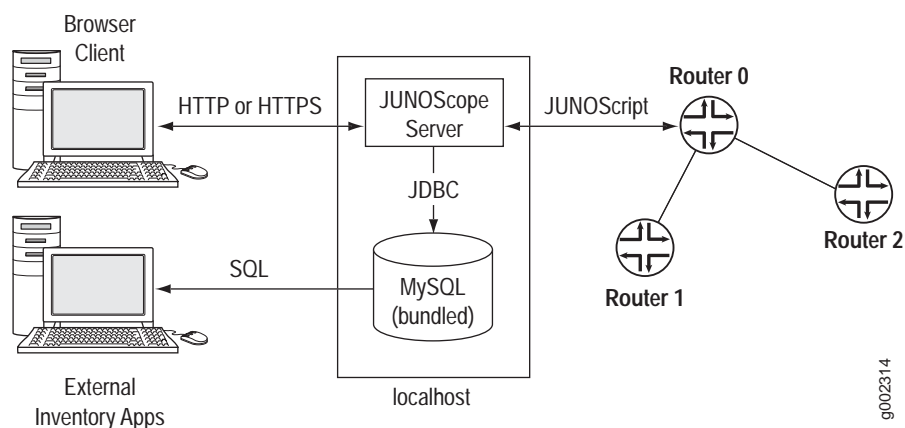
## Chapter 27

# Exporting Inventory Management System Data

This chapter is intended for users who are familiar with Relational Database Management Systems (RDBMS) and the Structured Query Language (SQL), who want to develop their own customer applications to access information from the JUNOScope Inventory Management System. It is not meant to be a complete SQL reference. Refer to your applicable SQL guides for complete reference information.

This chapter describes how an external inventory application can connect to the JUNOScope Inventory Management System database and extract Juniper Networks device inventory information by way of an SQL interface. The JUNOScope software is bundled with MySQL, an open source relational database management system (RDBMS). (See Figure 7.) The query examples used in this chapter to extract inventory data are for MySQL.

**Figure 7: Export Inventory Management System Data Topology Diagram**



External inventory applications can extract inventory data, such as hardware, software, licensed features, and inventory scan events, from the JUNOScope Inventory Management System database. A unique username and password must be configured during the JUNOScope software installation to enable read-only access to the Inventory Management System database.

JUNOScope versions 7.4 and higher support extracting Inventory Management System database information to an external inventory application.

This chapter also describes the SQL database schema to facilitate data export, and also describes each database table.

A demo Inventory Management System database is bundled with the JUNOScope installation, which consists of tables populated with sample inventory data. You can develop scripts or programs to practice extracting data from the demonstration database without having to scan real inventory data into the production Inventory Management System database. For more information about the demo tables, see “Demo Inventory Management System Database Tables” on page 302.

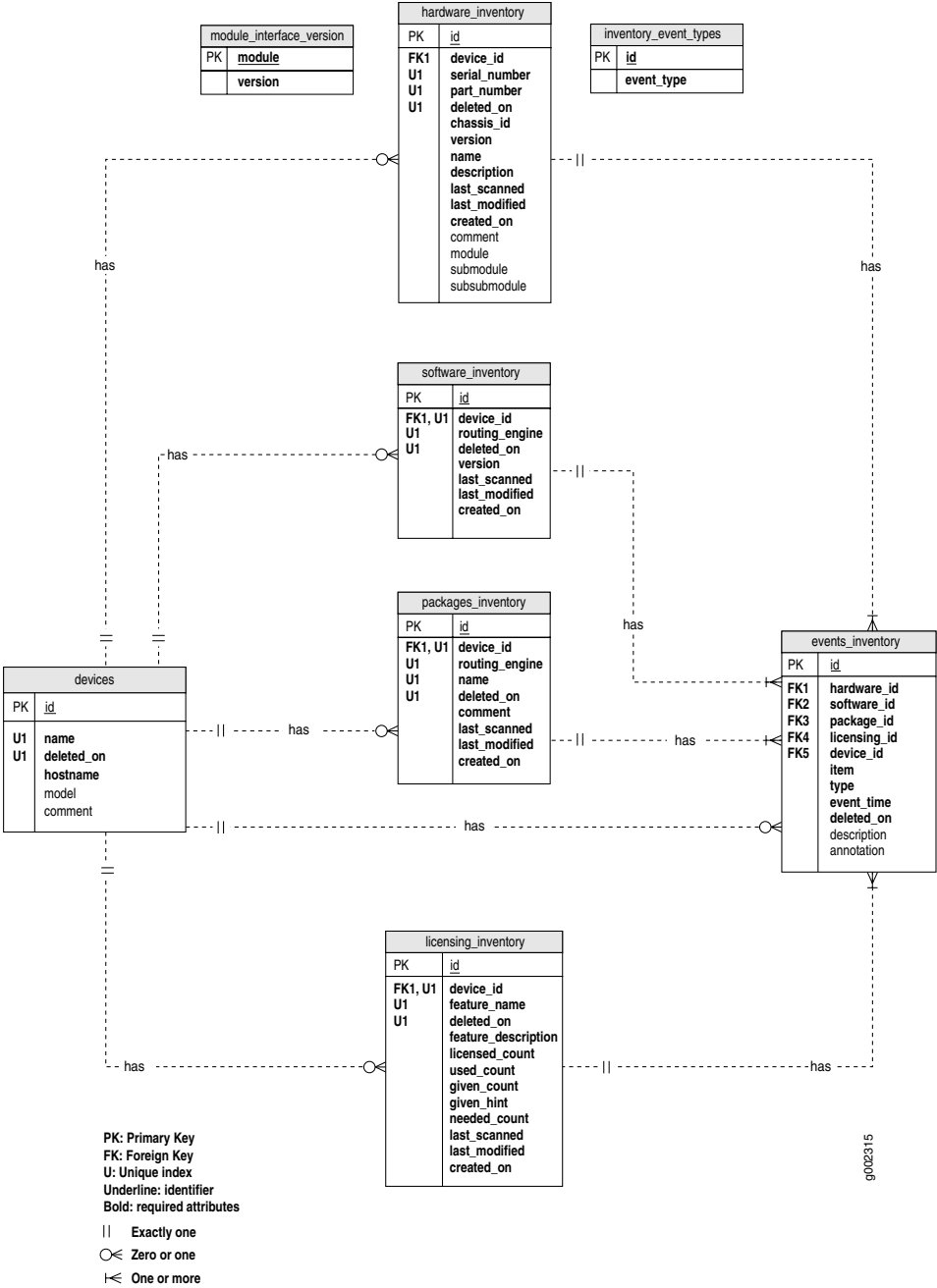
This chapter includes the following topics:

- Inventory Management System Database Entity Relationship on page 293
- Enabling Access to the Inventory Management System SQL Interface on page 294
- Changing the Username and Password and Creating Additional Users for the Inventory Management System SQL Interface on page 294
- Inventory Management System Database Tables on page 297
- Demo Inventory Management System Database Tables on page 302
- Demo Inventory Management System Reports on page 302
- Connecting to the Inventory Management System SQL Interface on page 303
- Querying All Hardware Inventory Items on page 304
- Querying All Hardware Inventory Items of a Device on page 305
- Querying JUNOS Software and Package Inventory Items on page 305
- Querying All Licensed Feature Inventory Data on page 306
- Querying Inventory Events Data on page 306

# Inventory Management System Database Entity Relationship

Figure 8 shows the JUNOScope Inventory Management System database entity relationship diagram.

Figure 8: Inventory Management System Database Entity Relationship Diagram



## Enabling Access to the Inventory Management System SQL Interface

The JUNOScope software installer can enable or disable access to the Inventory Management System SQL interface during the JUNOScope installation process.

After the JUNOScope software is installed, the installer can enable or disable the SQL interface to the Inventory Management System by running the reconfiguration script `jtk-setup.sh`.



**CAUTION:** Once the Inventory Management System SQL interface username and password have been added during JUNOScope software installation, the JUNOScope software will not prompt again to add them for reconfiguration.

## Changing the Username and Password and Creating Additional Users for the Inventory Management System SQL Interface

The database administrator can change the password of the database user already configured to access the Inventory Management System SQL interface. The database administrator can also create additional users to access the Inventory Management System SQL interface and grant the required privileges by using the sample SQL query below.

An SQL user with grant privileges (for example, root) can execute these statements:

```
-- grant read-only privilege to the inventory tables of @DBNAME@ for
@DBUSER_IMS@,
-- per the specification of the interface module 'core' and 'ims'.
-- @DBNAME@ is the name of the database, e.g. 'jtk' or 'demo'
-- @DBUSER_IMS@ is the name of the database user for the SQL interface to IMS
-- @DBUSER_IMS_PASSWORD@ is the password of the database user for the SQL
interface to IMS

GRANT SELECT (id) ON @DBNAME@.devices TO @DBUSER_IMS@@localhost IDENTIFIED BY
'@DBUSER_IMS_PASSWORD@';
GRANT SELECT (id) ON @DBNAME@.devices TO @DBUSER_IMS@"127.0.0.1" IDENTIFIED BY
'@DBUSER_IMS_PASSWORD@';
GRANT SELECT (id) ON @DBNAME@.devices TO @DBUSER_IMS@"@HOST_NAME@" IDENTIFIED
BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (id) ON @DBNAME@.devices TO @DBUSER_IMS@"@HOST_IP@" IDENTIFIED BY
'@DBUSER_IMS_PASSWORD@';
GRANT SELECT (id) ON @DBNAME@.devices TO @DBUSER_IMS@"%" IDENTIFIED BY
'@DBUSER_IMS_PASSWORD@';

GRANT SELECT (name) ON @DBNAME@.devices TO @DBUSER_IMS@@localhost IDENTIFIED BY
'@DBUSER_IMS_PASSWORD@';
GRANT SELECT (name) ON @DBNAME@.devices TO @DBUSER_IMS@"127.0.0.1" IDENTIFIED
BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (name) ON @DBNAME@.devices TO @DBUSER_IMS@"@HOST_NAME@" IDENTIFIED
BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (name) ON @DBNAME@.devices TO @DBUSER_IMS@"@HOST_IP@" IDENTIFIED
BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (name) ON @DBNAME@.devices TO @DBUSER_IMS@"%" IDENTIFIED BY
'@DBUSER_IMS_PASSWORD@';

GRANT SELECT (hostname) ON @DBNAME@.devices TO @DBUSER_IMS@@localhost IDENTIFIED
BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (hostname) ON @DBNAME@.devices TO @DBUSER_IMS@"127.0.0.1"
```

```

IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (hostname) ON @DBNAME@.devices TO @DBUSER_IMS@@"@HOST_NAME@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (hostname) ON @DBNAME@.devices TO @DBUSER_IMS@@"@HOST_IP@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (hostname) ON @DBNAME@.devices TO @DBUSER_IMS@@"%" IDENTIFIED BY
 '@DBUSER_IMS_PASSWORD@';

GRANT SELECT (model) ON @DBNAME@.devices TO @DBUSER_IMS@@"localhost IDENTIFIED BY
 '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (model) ON @DBNAME@.devices TO @DBUSER_IMS@@"127.0.0.1" IDENTIFIED
 BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (model) ON @DBNAME@.devices TO @DBUSER_IMS@@"@HOST_NAME@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (model) ON @DBNAME@.devices TO @DBUSER_IMS@@"@HOST_IP@" IDENTIFIED
 BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (model) ON @DBNAME@.devices TO @DBUSER_IMS@@"%" IDENTIFIED BY
 '@DBUSER_IMS_PASSWORD@';

GRANT SELECT (comment) ON @DBNAME@.devices TO @DBUSER_IMS@@"localhost IDENTIFIED
 BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (comment) ON @DBNAME@.devices TO @DBUSER_IMS@@"127.0.0.1"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (comment) ON @DBNAME@.devices TO @DBUSER_IMS@@"@HOST_NAME@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (comment) ON @DBNAME@.devices TO @DBUSER_IMS@@"@HOST_IP@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (comment) ON @DBNAME@.devices TO @DBUSER_IMS@@"%" IDENTIFIED BY
 '@DBUSER_IMS_PASSWORD@';

GRANT SELECT (deleted_on) ON @DBNAME@.devices TO @DBUSER_IMS@@"localhost
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (deleted_on) ON @DBNAME@.devices TO @DBUSER_IMS@@"127.0.0.1"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (deleted_on) ON @DBNAME@.devices TO @DBUSER_IMS@@"@HOST_NAME@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (deleted_on) ON @DBNAME@.devices TO @DBUSER_IMS@@"@HOST_IP@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT (deleted_on) ON @DBNAME@.devices TO @DBUSER_IMS@@"%" IDENTIFIED BY
 '@DBUSER_IMS_PASSWORD@';

GRANT SELECT ON @DBNAME@.module_interface_version TO @DBUSER_IMS@@"localhost
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.module_interface_version TO @DBUSER_IMS@@"127.0.0.1"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.module_interface_version TO @DBUSER_IMS@@"@HOST_NAME@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.module_interface_version TO @DBUSER_IMS@@"@HOST_IP@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.module_interface_version TO @DBUSER_IMS@@"%" IDENTIFIED
 BY '@DBUSER_IMS_PASSWORD@';

GRANT SELECT ON @DBNAME@.hardware_inventory TO @DBUSER_IMS@@"localhost IDENTIFIED
 BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.hardware_inventory TO @DBUSER_IMS@@"127.0.0.1"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.hardware_inventory TO @DBUSER_IMS@@"@HOST_NAME@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.hardware_inventory TO @DBUSER_IMS@@"@HOST_IP@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.hardware_inventory TO @DBUSER_IMS@@"%" IDENTIFIED BY
 '@DBUSER_IMS_PASSWORD@';

```

```

GRANT SELECT ON @DBNAME@.software_inventory TO @DBUSER_IMS@@localhost IDENTIFIED
BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.software_inventory TO @DBUSER_IMS@"127.0.0.1"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.software_inventory TO @DBUSER_IMS@"@HOST_NAME@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.software_inventory TO @DBUSER_IMS@"@HOST_IP@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.software_inventory TO @DBUSER_IMS@"%" IDENTIFIED BY
 '@DBUSER_IMS_PASSWORD@';

```

```

GRANT SELECT ON @DBNAME@.packages_inventory TO @DBUSER_IMS@@localhost IDENTIFIED
BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.packages_inventory TO @DBUSER_IMS@"127.0.0.1"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.packages_inventory TO @DBUSER_IMS@"@HOST_NAME@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.packages_inventory TO @DBUSER_IMS@"@HOST_IP@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.packages_inventory TO @DBUSER_IMS@"%" IDENTIFIED BY
 '@DBUSER_IMS_PASSWORD@';

```

```

GRANT SELECT ON @DBNAME@.licensing_inventory TO @DBUSER_IMS@@localhost
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.licensing_inventory TO @DBUSER_IMS@"127.0.0.1"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.licensing_inventory TO @DBUSER_IMS@"@HOST_NAME@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.licensing_inventory TO @DBUSER_IMS@"@HOST_IP@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.licensing_inventory TO @DBUSER_IMS@"%" IDENTIFIED BY
 '@DBUSER_IMS_PASSWORD@';

```

```

GRANT SELECT ON @DBNAME@.events_inventory TO @DBUSER_IMS@@localhost IDENTIFIED
BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.events_inventory TO @DBUSER_IMS@"127.0.0.1" IDENTIFIED
BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.events_inventory TO @DBUSER_IMS@"@HOST_NAME@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.events_inventory TO @DBUSER_IMS@"@HOST_IP@" IDENTIFIED
BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.events_inventory TO @DBUSER_IMS@"%" IDENTIFIED BY
 '@DBUSER_IMS_PASSWORD@';

```

```

GRANT SELECT ON @DBNAME@.inventory_event_types TO @DBUSER_IMS@@localhost
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.inventory_event_types TO @DBUSER_IMS@"127.0.0.1"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.inventory_event_types TO @DBUSER_IMS@"@HOST_NAME@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.inventory_event_types TO @DBUSER_IMS@"@HOST_IP@"
IDENTIFIED BY '@DBUSER_IMS_PASSWORD@';
GRANT SELECT ON @DBNAME@.inventory_event_types TO @DBUSER_IMS@"%" IDENTIFIED BY
 '@DBUSER_IMS_PASSWORD@';

```

## Inventory Management System Database Tables

The Inventory Management System database user has read-only access or **SELECT** privileges to the following JUNOScope Inventory Management System database tables:

- devices Table on page 297
- hardware\_inventory Table on page 298
- software\_inventory Table on page 298
- packages\_inventory Table on page 299
- licensing\_inventory Table on page 299
- events\_inventory Table on page 300
- inventory\_events\_types Table on page 300
- module\_interface\_version Table on page 301

The database user does not have access to other JUNOScope database tables that are not described in this document.

### devices Table

The `devices` database table stores all Juniper Networks devices added in JUNOScope. For information about adding devices, see “Setting Up Devices” on page 59. Table 31 shows the fields and columns in the `devices` table.

**Table 31: devices Table**

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI		auto_increment
name	varchar(40)				
hostname	varchar(40)				
model	varchar(20)	Yes			
comment	text	Yes			
deleted_on	timestamp(14)	Yes			

**hardware\_inventory Table**

The hardware\_inventory database table stores information about all hardware components installed in devices added in JUNOScope. Table 32 shows the fields and columns in the hardware\_inventory table.

**Table 32: hardware\_inventory Table**

Field	Type	Null	Key	Default	Extra
id	int(11)	PRI			auto_increment
device_id	int(11)			0	
chassis_id	varchar(20)				
version	varchar(20)				
part_number	varchar(20)				
serial_number	varchar(20)				
name	varchar(40)				
module	varchar(40)	Yes			
submodule	varchar(40)	Yes			
subsubmodule	varchar(40)	Yes			
description	varchar(40)				
comment	text	Yes			
last_scanned	timestamp(14)	Yes			
last_modified	timestamp(14)	Yes			
created_on	timestamp(14)	Yes			
deleted_on	timestamp(14)	Yes			

**software\_inventory Table**

The software\_inventory database table stores information about all JUNOS software packages installed on devices added in JUNOScope. Table 33 shows the fields and columns in the software\_inventory table.

**Table 33: software\_inventory Table**

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI		auto_increment
device_id	int(11)			0	
routing_engine	varchar(10)				
version	varchar(20)				
last_scanned	timestamp(14)	Yes			
last_modified	timestamp(14)	Yes			
created_on	timestamp(14)	Yes			
deleted_on	timestamp(14)	Yes			

***packages\_inventory Table***

The `packages_inventory` database table stores all JUNOS software packages installed on devices added in JUNOScope. Table 34 shows the fields and columns in the `packages_inventory` table.

**Table 34: packages\_inventory Table**

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI		auto_increment
device_id	int(11)			0	
routing_engine	varchar(10)				
name	varchar(20)				
comment	varchar(100)				
last_scanned	timestamp(14)	Yes			
last_modified	timestamp(14)	Yes			
created_on	timestamp(14)	Yes			
deleted_on	timestamp(14)	Yes			

***licensing\_inventory Table***

The `licensing_inventory` database table stores information about all licensed features installed on devices added in JUNOScope. Table 35 shows the fields and columns in the `licensing_inventory` table.

**Table 35: licensing\_inventory Table**

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI		auto_increment
device_id	int(11)			0	
feature_name	varchar(64)				
feature_description	varchar(64)				
licensed_count	int(11)			0	
used_count	int(11)			0	
given_count	int(11)			0	
given_hint	varchar(64)				
needed_count	int(11)			0	
last_scanned	timestamp(14)	Yes			
last_modified	timestamp(14)	Yes			
created_on	timestamp(14)	Yes			
deleted_on	timestamp(14)	Yes			

**events\_inventory Table**

The `events_inventory` database table stores all events that occur during an inventory scan using the Inventory Management System. For more information about running an inventory scan, see “Scanning Inventory Data” on page 231. Table 36 shows the fields and columns in the `events_inventory` table.

**Table 36: events\_inventory Table**

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI		auto_increment
device_id	int(11)			0	
hardware_id	int(11)			0	
software_id	int(11)			0	
package_id	int(11)			0	
licensing_id	int(11)			0	
item	varchar(40)				
type	varchar(10)				
description	text	Yes			
annotation	text	Yes			
event_time	timestamp(14)	Yes			
deleted_on	timestamp(14)	Yes			

**inventory\_events\_types Table**

The `inventory_events_types` database table stores all inventory scan event types. Table 37 shows the fields and columns in the `inventory_events_types` table.

**Table 37: inventory\_events\_types Table**

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI		auto_increment
event_type	varchar(10)				

### ***module\_interface\_version Table***

The `module_interface_version` database table stores and identifies the module name and version of the Inventory Management System database. Table 38 shows the fields and columns in the `module_interface_version` table.

**Table 38: `module_interface_version` Table**

Field	Type	Null	Key	Default	Extra
module	varchar(40)		PRI		
version	int(11)			0	

For example, the `module` field can be either:

```
module = core, version = 1
module = ims, version = 1
```

The `version` field is incremented when an update is made to the documented SQL interface with respect to the module. An update occurs when:

- A new public column is added.
- The meaning of a public column changes.
- A public column is removed.

The module version identifies the public subset of the JUNOScope database schema.

### **Table-to-Module Mapping**

Two modules are currently defined: *ims* and *core*.

#### ***ims Module***

Version 1 of the *ims* module consists of the following tables:

- `hardware_inventory` Table on page 298
- `software_inventory` Table on page 298
- `packages_inventory` Table on page 299
- `licensing_inventory` Table on page 299
- `events_inventory` Table on page 300
- `inventory_events_types` Table on page 300

#### ***core Module***

The *core* module (version 1) consists of the `devices` table listed in Table 31 on page 297.

## Demo Inventory Management System Database Tables

---

The demo inventory tables are populated with sample data. Use the demo inventory tables to:

- Experiment and execute the sample query by way of the external SQL interface.
- See and generate sample reports based on the demo data. You can experiment and use Inventory Management System reports without scanning devices on the network.

The demo tables are separate from the normal JUNOScope tables, and belong to a different database; **demo** rather than **jtk**. The demo tables are created during the JUNOScope Inventory Management System software installation.

The following demo tables, similar to those described in “Inventory Management System Database Entity Relationship” on page 293, are created and populated with sample data.

- `demo.devices`
- `demo.hardware_inventory`
- `demo.software_inventory`
- `demo.packages_inventory`
- `demo.licensing_inventory`
- `demo.events_inventory`
- `demo.inventory_event_types`
- `demo.module_interface_version`

## Demo Inventory Management System Reports

---

Several demo custom reports are packaged as part of the JUNOScope installation. Sample custom reports installed include:

- Licensing summary
- Sorted hardware summary
- All inventory changes in the network between Friday and Sunday
- All Gigabit Ethernet PICs
- All JUNOS Release 7.0 or higher

## Connecting to the Inventory Management System SQL Interface

---

To connect to the Inventory Management System database using the MySQL client provided in the JUNOScope installation, type the following:

```
% $JTK_INSTALL/mysql/bin/mysql
  - -socket=$JTK_INSTALL/data/db/mysql.sock
  - -port=$DBPORT
  - -host=$HOSTNAME
  - -user=$DBUSER_IMS
  - -password=$DBUSER_IMS_PASSWORD $DBNAME
```

Where:

- `$JTK_INSTALL` is the path of the JUNOScope installation.
- `$DBPORT` is the port number of the database connection. The default port number is 3306.
- `$HOSTNAME` is the hostname of database server.
- `$DBUSER_IMS` is the database user for read-only access to the Inventory Management System. This is the username that the JUNOScope administrator provided during the installation process.
- `$DBUSER_IMS_PASSWORD` is the password for the Inventory Management System database user. This is the password that the JUNOScope administrator provided during the installation process.
- `$DBNAME` is the database name `jtk` for accessing the production Inventory Management System database, or `demo` for accessing the demonstration database.

## Querying All Hardware Inventory Items

---

All of the database query examples are specific to MySQL. If the underlying database is not MySQL, the query will be different.

To extract all active hardware inventory items stored in the Demo Inventory Management System database, use the following query:

```
mysql>SELECT dev.name,
           dev.model,
           hw.name,
           hw.version,
           hw.part_number,
           hw.serial_number,
           hw.description,
           hw.chassis_id,
           hw.module,
           hw.submodule,
           hw.subsubmodule,
           hw.last_scanned,
           hw.created_on
        FROM demo.hardware_inventory hw
       INNER JOIN demo.devices dev
              ON hw.device_id = dev.id
          WHERE hw.deleted_on = 0 AND dev.deleted_on = 0;
```

When an item is no longer active, the `deleted_on` field is updated with the time the entry was removed.

You can make an entry inactive in one of two ways:

- When the item is no longer part of the network (for example, a PIC is removed from a chassis).
- The row is administratively removed from the database by marking the `deleted_on` field to non-zero, making it virtually hidden from the user. This is not done by user action, but by manually setting the `deleted_on` field in the database.

The `last_scanned` field stores the timestamp of the item (row) when it was last updated and processed by the Inventory Management System, regardless of whether the item (row) itself was modified.

The `created_on` field stores the timestamp of the item (row) when it was first scanned and processed by the Inventory Management System. If this item is removed from one chassis and moved to another, the `created_on` timestamp will remain the same.

## Querying All Hardware Inventory Items of a Device

---

To extract all active hardware inventory items of device XYZ, use the following query:

```
mysql> SELECT dev.name,
              dev.model,
              hw.name,
              hw.version,
              hw.part_number,
              hw.serial_number,
              hw.description,
              hw.chassis_id,
              hw.module,
              hw.submodule,
              hw.subsubmodule,
              hw.last_scanned,
              hw.created_on
          FROM demo.hardware_inventory hw
         INNER JOIN demo.devices dev
              ON hw.device_id = dev.id
          WHERE hw.deleted_on = 0
              AND dev.deleted_on = 0
              AND dev.name = 'XYZ';
```

## Querying JUNOS Software and Package Inventory Items

---

To extract all active software and package inventory items stored in the Inventory Management System, use the following query:

```
mysql> SELECT dev.name,
              dev.model,
              sw.routing_engine,
              sw.version,
              pkg.name,
              pkg.comment,
              sw.last_scanned,
              sw.created_on
          FROM demo.software_inventory sw
         INNER JOIN demo.packages_inventory pkg
              ON sw.device_id = pkg.device_id
         INNER JOIN demo.devices dev
              ON sw.device_id = dev.id
          WHERE sw.routing_engine = pkg.routing_engine
              AND sw.deleted_on = 0
              AND pkg.deleted_on = 0
              AND dev.deleted_on = 0;
```

## Querying All Licensed Feature Inventory Data

---

To extract all active licensed features inventory items stored in the Inventory Management System database, use the following query:

```
mysql> SELECT dev.name,
              dev.model,
              lic.feature_name,
              lic.feature_description,
              IF(given_count, given_count, "") AS given,
              lic.used_count,
              lic.licensed_count,
              lic.needed_count,
              lic.last_scanned,
              lic.created_on
          FROM demo.licensing_inventory lic
         INNER JOIN demo.devices dev ON lic.device_id = dev.id
        WHERE lic.deleted_on = 0 AND dev.deleted_on = 0;
```

## Querying Inventory Events Data

---

To extract all active inventory events stored in the Inventory Management System database, use the following query:

```
mysql> SELECT ev.type,
              ev.item,
              ev.event_time,
              ev.description AS details,
              dev.name,
              IF(ev.hardware_id != 0, hw.serial_number, "")
              AS serial_number,
              IF(ev.hardware_id != 0, hw.description,
                IF(ev.software_id != 0, "",
                  IF(ev.package_id != 0, pkg.name,
                    IF(ev.licensing_id != 0,
                      lic.feature_description, ""))))
              AS description,
              IF(ev.software_id != 0, sw.routing_engine,
                IF(ev.package_id != 0, pkg.routing_engine, ""))
              AS routing_engine
          FROM demo.events_inventory ev
         INNER JOIN demo.devices dev ON ev.device_id = dev.id
         LEFT JOIN demo.hardware_inventory hw ON ev.hardware_id = hw.id
         LEFT JOIN demo.software_inventory sw ON ev.software_id = sw.id
         LEFT JOIN demo.packages_inventory pkg ON ev.package_id = pkg.id
         LEFT JOIN demo.licensing_inventory lic
           ON ev.licensing_id = lic.id
        WHERE ev.deleted_on = 0;
```