eG Integration With Trouble Ticketing Systems

eG Enterprise v6

Restricted Rights Legend

Trademarks

Copyright

Table of Contents

Table of Figures

1

Introduction

eG Enterprise includes extensive monitoring capabilities for IT Infrastructure components. Problems detected by eG products can be reported to users in various ways – via the web, over email, and via SNMP traps to any SNMP console. Many enterprises use **Trouble Ticketing** (TT) systems to track problems with their IT infrastructures. Besides tracking the current problems, a trouble ticketing system enables an operator to dispatch service requests to the appropriate maintenance personnel. Maintenance personnel can use the trouble ticket system to update and monitor the status of current problems and follow these through to final resolution.

The integration of eG Enterprise with TT systems facilitates the following actions to be automatically performed in the TT system based on the open alarms in eG Enterprise:

- trouble tickets to be opened in the TT systems as and when a new alarm is detected by eG Enterprise;
- trouble tickets to be modified as and when an existing alarm is modified in eG Enterprise;
- trouble tickets to be closed as and when an alarm is removed in eG Enterprise;

1.1 How the eG Enterprise to TT System Integration Works

1.1.1 Alarms in eG Enterprise

To understand how the integration of the eG manager with a trouble ticketing system works, lets first consider what is an alarm. An alarm in eG Enterprise, is identified by an **Alarm ID**. At any given instant of time, an **Alarm ID** is a unique combination of the following attributes:

a. The problem component-type

b. The problem component (i.e., network device, application, etc.)

c. The problem layer

d. The problem priority (Critical, Major, Minor)

The eG monitoring interface lists alarms that currently exist in the eG Enterprise system. The goal of the eG Enterprise integration with TT systems is to be able to forward updated information on current alarms to the TT system.

Every time there is a state change (e.g., change of priority or correction of a problem) detected in the monitored environment, the eG manager checks the combination of component, component-type, layer, and priority combination for all open problems with their previous values to determine whether a new alarm has been generated, an existing alarm has been modified, or whether an existing alarm has been closed. If a new alarm has been generated, the eG manager assigns a distinct alarm ID for this alarm. If an existing alarm has been modified or closed, the eG manager retains the earlier assigned alarm ID for this alarm. Modification of an alarm can include any of the following cases:

- A change in the alarm priority: This could be a switch to a higher or lower priority.

- A change in the alarm description: For example, originally, a usage-related alarm may have been raised on disk 'D' of a server. Later, disk 'C' of the same server might have experienced a space crunch, causing another alarm to be raised. In this case, the description of the original alarm will change to indicate that both disks C and D are experiencing a problem, but the alarm ID will not change. Changes in alarm description may also happen if additional tests being run for the same layer indicate a problem. A change may involve either an addition to the description (as in the example above) or a removal of one or more descriptors (e.g., the space usage of disk 'C' in the example above returning to a normal condition).

- A change in the list of impacted services

Each alarm is associated with a start date and time. The start date and time signifies when the alarm was first generated by the eG manager. Any change in the state of the alarm during a subsequent time does not cause a change in the start date and time of the alarm. Hence, even if an alarm changes in priority at a later time, its start date and time remain the same, until the alarm is finally closed. When an alarm is closed, a normal alert is generated, which will bear the current date and time.

In order to avoid conflicts/duplication of alarm IDs generated by each of the managers in a redundant eG manager cluster, the alarm ID is expressed as a string that is of the form ***<eG_Manager>_<numeric_value>***, where the ***<numeric_value>*** is a timestamp of when the alarm was first generated.

Prior to generating an alarm, the eG managers in a cluster synchronize with each other to ensure that duplicate alarms are not generated or that different alarm IDs are not generated for the same problem. As in the case of email alerts and SNMP traps, each manager in the cluster is responsible for generating alarms for agents that are directly reporting to the manager.

1.1.2 Integration with Trouble Ticketing Systems

The eG manager can be configured so that whenever an alarm undergoes a change – either generation, modification, or closure – the manager communicates this information to a TT system.

This communication can be in any of the forms mentioned below:

- In recent times, many trouble ticketing systems have been found to embed a unique mail interface that receives email alerts of problems in the environment. The eG Enterprise system can be configured to use this interface to send alarms generated by the eG manager as email alerts to the trouble ticketing system. Based on the mails so received, the trouble ticketing system may generate trouble tickets and forward them to the concerned maintenance personnel.

- The eG manager can also send its alarms as SNMP traps to third-party SNMP management systems. When doing so, you can specifically configure the eG manager to send these traps as trouble tickets to the third-party system. This ensures that every SNMP trap sent by the eG manager is tagged with a unique TT ID, which helps track the status of the problem for which the trap was originally raised.

- The eG Manager supports a command line interface, that can be configured to automatically execute TT system-specific commands as and when alarms are added, modified, or deleted in eG Enterprise. This interface offers a way of communication between the eG Manager and a TT system.

- The eG Manager can also forward its alarm information to any web services interface that the Trouble Ticketing System may support to trigger the automatic creation/closure (as the case may be) of trouble tickets.

Each chapter in this document discusses each of these integration modes in detail.

1.1.3 Handling eG Alarms in a Trouble Ticketing System

A trouble ticket system must be configured to process alarms reported to it by an eG manager. The alarm ID must be used to uniquely identify an alarm. The functions that the TT system must perform are:

- Determine if an alarm ID indicates a new alarm. If yes, open a new trouble ticket.
- If an alarm ID indicates an existing alarm, check the priority of the alarm. If the priority is **Normal**, this implies that the alarm has been closed in eG Enterprise. Hence, close the corresponding trouble ticket in the TT system.
- If an alarm ID indicates an existing alarm and the priority of the alarm is not **Normal**, update the corresponding trouble ticket with the current priority of the alarm and with its current description.

These functions often involve scripting/configurations on the TT system.

Once the above steps are accomplished, by reviewing the status of the trouble tickets, administrators can be immediately aware of the current status of the infrastructure being monitored, without having to login to the eG Enterprise console.

- If a standalone (i.e., non-redundant) eG manager is restarted, all outstanding alarms and hence, all open trouble tickets will be closed. After the restart, if an old problem re-occurs, the restarted manager will assign a new alarm ID to this problem; as a result, new trouble tickets will be opened for such problems.

- In a redundant configuration, when a manager is restarted, it checks if the other manager is available. If the other manager in the cluster is not available, all outstanding alarms will be closed. On the other hand, if the other manager in the cluster is available, then the manager being restarted will synchronize alarm information with the other manager. When it detects a problem, the restarted manager checks to see if the other manager in the cluster has already assigned an alarm ID to this problem. If so, then the restarted manager assigns the same ID to the problem. In such a case, new trouble tickets will not be opened for the existing problems.

- In rare instances, when there are rapid alarm transitions (eg., from critical to normal to critical state) for the same component type-component name-layer-priority combination in a redundant eG manager configuration, the same alarm ID may be re-used to refer to the new alarm.

- The same eG manager can be configured to different modes of integration with a TT system – be it email integration, command line integration, or web services-based integration.

- Since the eG manager forwards the current status of an alarm to the TT system, and since such transmission is done only at periodic intervals, the eG Enterprise-TT system integration does not capture all state transitions in the infrastructure being monitored. For instance, if the **MailCheckPeriod** setting is 3 mins, an event that happens and gets corrected within 1 min is never captured in the TT system. Consider changing the **MailCheckPeriod** setting to a lower value (upto 1sec), if you require higher sensitivity in trouble ticket tracking. Obviously, lower the value of the **MailCheckPeriod**, greater is the overhead on the eG manager.

1.2 Pre-requisites for Integrating eG Enterprise with a TT System

Prior to integrating with any TT system, ensure the following are in order:

- The eG license should enable the **Trouble Ticket Integration** capability of the eG manager.

- In case of a redundant manager configuration, all the eG managers that are part of the cluster should have the trouble ticketing integration capability enabled in their respective licenses.

- The basic settings for the integration should be defined. Section 1.2.1 describes how to configure these

basic settings.

1.2.1 Configuring the Basic Settings for the Trouble Ticket Integration

Before attempting the TT integration, you need to indicate the following:

- What alarms should be forwarded to the TT system?
- Should detailed diagnostics be sent to the TT system?
- Should normal alerts be routed to the TT system?
- Should logging be enabled? If so, what should be the maximum size of the log file?

To configure the above, follow the steps outlined below:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 1.1 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 1.1, expand the **Trouble Ticket Integration** node. Then select the **Common Settings** sub-node. A **Common Settings** section will open in the right panel, as depicted by Figure 1.1.

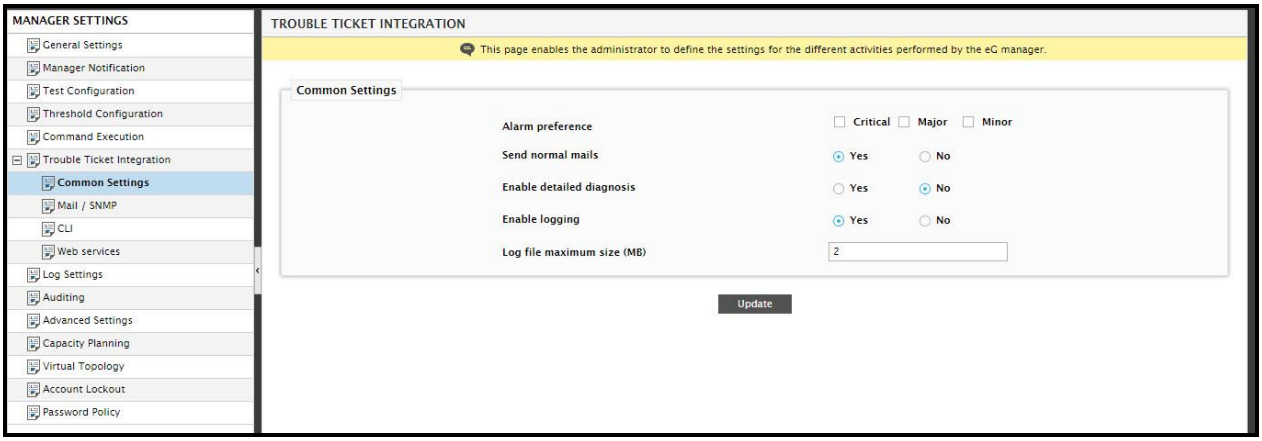

Figure 1.1: Configuring the common integration settings

4. From this section, first select the type of alarms that the eG manager should send to the TT system after the integration. For this, select the **Critical**, **Major**, and/or **Minor** check boxes against **Alarm preferences**.
5. If you want the TT system to be intimated if a problem is resolved, so that the corresponding trouble ticket is automatically closed, then set the **Send normal alerts** flag to **Yes**.
6. To ensure that detailed diagnostics, if available, are sent along with alarm information to the TT system, set the **Send detailed diagnostics** flag to **Yes**.

Even if this flag is set to **Yes**, the eG manager will send detailed diagnostics to the TT system, only if the following conditions are fulfilled:

- The **Detailed Diagnosis** capability should be enabled in the eG license.
- In case of a redundant manager configuration again, all the eG managers that are part of the cluster should have the detailed diagnosis capability enabled in their respective licenses.
- Both the normal and abnormal frequencies configured globally or at the test-level for collecting detailed measures should not be 0.
- The **DETAILED DIAGNOSIS** flag at the test-level, should be set to **On**.
- The eG manager should integrate with the TT system via the TT mail interface or the TT CLI only.

7. Next, if you want the eG manager to track the alarms routed to the TT system and the errors, successes, and failures in the process, you can optionally configure the eG manager to maintain logs of the same. To trigger log generation, set the **Enable logging** flag to **Yes**.

- Regardless of the status of this flag, logs will not be generated if eG alarms are transmitted to the TT system via the TT mail interface.
- Where the eG manager integrates with a TT system via the TT CLI, setting this flag to **Yes** will result in the automatic creation of a **ttexec.log** file in the **<EG_INSTALL_DIR>\manager\logs** directory. By default, these log files capture both the errors and the standard output of the specified **Command**. A sample log is provided below, where both standard output and errors have been logged:

```
03/11/2009 18:25:03 INFO AlarmId
"192.168.10.133_1257252903826" -DATE "Nov 03, 2009" -TIME
"18:25:02" -Priority "Critical" -ComponentType "Generic" -
ComponentName "gen133:NULL" -Layer "Application Processes" -
Desc "-|Processes|Process not running{cmd}|gen133" -Service
"-"#AlarmId "192.168.10.133_1257252903842" -DATE "Nov 03,
2009" -TIME "18:25:02" -Priority "Major" -ComponentType
"Generic" -ComponentName "gen133:NULL" -Layer "Application
Processes" -Desc "-|Processes|Process not
running{notepad}|gen133" -Service "-"
03/11/2009 18:36:12 ERROR Executing command - ech AlarmId
"192.168.10.133_1257253386239" -DATE "Nov 03, 2009" -TIME
"18:32:37" -Priority "Normal" -ComponentType "Generic" -
ComponentName "gen133:NULL" -Layer "Application Processes" -
Desc "-|Processes|Process not running{cmd}|gen133" -Service
"-"#AlarmId "192.168.10.133_1257253386255" -DATE "Nov 03,
2009" -TIME "18:32:38" -Priority "Normal" -ComponentType
"Generic" -ComponentName "gen133:NULL" -Layer "Application
Processes" -Desc "-|Processes|Process not
running{notepad}|gen133" -Service "-"
03/11/2009 18:36:12 ERROR 'ech' is not recognized as an
internal or external command,
```

- Where the eG manager integrates with a TT system via a web services framework, setting this flag to **Yes** will result in the automatic creation of **<TT_System_Name>_log** file in the **<EG_INSTALL_DIR>\manager\logs** directory. By default, these log files capture both the errors and the standard output.
- This flag cannot be used to enable logging of SNMP trap transmissions to SNMP managers. Where TT integration is enabled over SNMP traps, logging should be enabled using the procedure discussed in Section 3.3.

8. Once the **Enable Logging** flag is set to **Yes**, you will be allowed to indicate the maximum size (in MB) up to which the log file should be allowed to grow. The default size is 2 MB.

- Where the eG manager uses the TT CLI to integrate with a TT system, if the **ttexec.log** file grows beyond the size limit configured in the **Log file maximum size** text box, the details originally logged in the **ttexec.log** file will be moved to another log file named **ttexec.log.1**, and the newer information will be logged in the **ttexec.log** file. This log roll over mechanism helps ensure that the log file does not grow beyond control.
- Where the eG manager uses the web services framework to integrate with a TT system, if the **<TT_System_Name>_log** file grows beyond the size limit configured in the **Log file maximum size** text box, the details originally logged in the **<TT_System_Name>_log** file will be moved to another log file named **<TT_System_Name>_log.1**, and the newer information will be logged in the **<TT_System_Name>_log** file. This roll over will continue until a maximum of 10 log files are created – i.e., until a log file named **<TT_System_Name>_log.9** is created. After this point, the eG manager will automatically delete the **<TT_System_Name>_log.9** file, as it contains the oldest logs.

9. Finally, click the the **Update** button in Figure 1.1 to save the settings.

2

Trouble Ticket Integration Using the TT Mail Interface

The eG manager can be configured so that whenever an alarm undergoes a change – either generation, modification, or closure – the manager communicates this information to a TT system. This communication can be in the form of formatted email messages that can be processed by a TT system using email interfaces that it supports.

This chapter discusses how eG integrates with a TT system that supports an email interface.

2.1 Pre-requisites for Integrating with a TT System via a TT Mail Interface

Before configuring an eG manager to integrate with a TT system via its email interface, make sure that the eG manager has been configured with a **Mail server** and an **Admin mail ID**.

Refer to the *Administering the eG Enterprise Suite* document to know how to configure the mail server.

2.2 Integrating the eG Manager with a TT System via a TT Mail Interface

To achieve this, follow the steps below:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 2.1 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 2.1, expand the **Trouble Ticket Integration** node. Then, select the **Mail / SNMP** sub-node. A **Mail and SNMP** section will open in the right panel, as depicted by Figure 2.1.

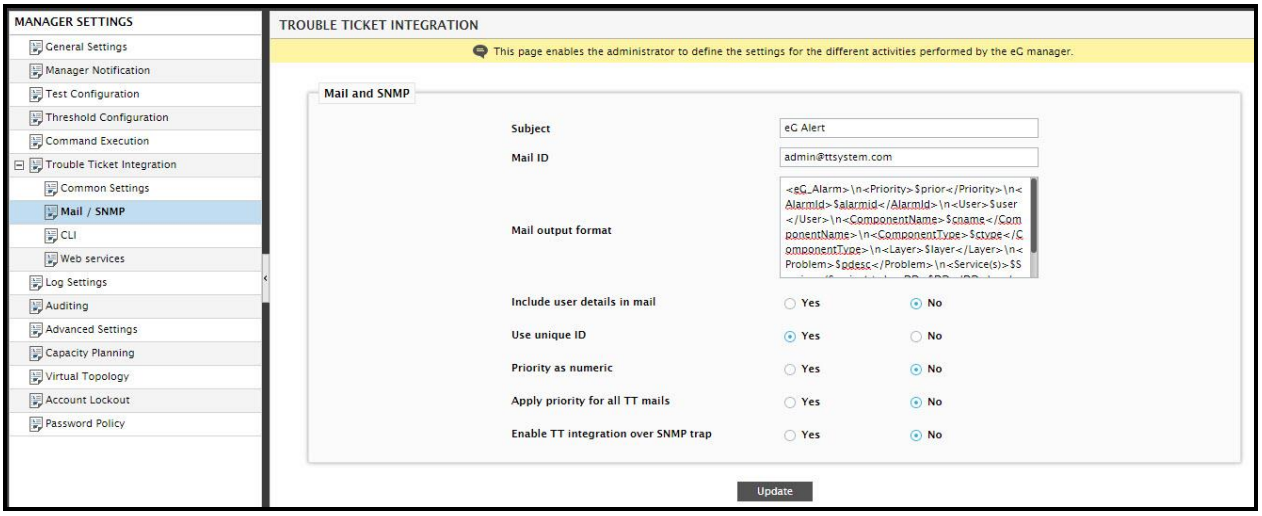

Figure 2.1: Configuring eG integration with a TT system via an email interface

4. In the right panel, first specify the **Subject** of the email alerts that the eG manager will send to the TT system. To ensure that the mail subject reflects the problem component name, problem component type, the problem priority, etc., variables can be used in the **Subject** definition, in the following format: *$alarmid#$user#$cname#$ctype#$layer#$prior#$pdesc*. The variable *$alarmid* in the **Subject** ensures that the subject of the TT mail contains the alarm id. Similarly, the *$user, $cname, $ctype* variables represent the user with whom the alarm is associated, the name of the problem component, and the problem component type, respectively. Likewise, the *$layer, $prior, and $pdesc* variables denote the problem layer, the alarm priority, and the alarm description, respectively. The '#' is the separator, but it can be changed. You can rearrange the order of the variables, and even omit a few variables if need be; but, the variable names and the '$' symbol preceding the name should not be changed. A sample subject has been provided below:

192.168.10.12_14678945321#john#printer:NULL#Host_system#NETWORK#Critical#Network connection down

This mail subject has been described below:

- **192.168.10.12_14678945321** is the **alarm id** that eG's trouble ticketing engine automatically generates every time a new alarm is processed by it; typically, this will be of the format: *<ManagerIP>_<a long value>*
- **john** is the user with whom the problem component is associated
- **printer:NULL** is the hostname of the problem component
- **Host_system** is the component type
- **NETWORK** is the layer name
- **Critical** is the alarm priority
- **Network connection down** is the alarm description

Note that a mail **Subject** specification characterized by variables discussed above will work only if the **SeparateMails** flag in the **[TTMAIL]** section of the **eg_services.ini** file (in the **<EG_INSTALL_DIR>\manager\config** directory) is switched on. By default, this flag is set to **No**, indicating that a single TT mail will comprise of details pertaining to all the alarms that were raised by the eG Enterprise system during that point in time. In such a case, the variables in the **Subject** cannot be substituted by the corresponding problem information; in this case therefore, by default, **eGTTMail** will appear as the subject of the TT mail. However, if every problem event in the environment should generate a separate TT mail, then set the **SeparateMails** flag to **Yes**. In such a case, the variables in the **Subject** will be substituted by the corresponding details from the problem information.

5. Next, against the **Mail ID** field, provide a comma-separated list of email IDs to which the email alerts are to be delivered.

6. Then, specify the **Mail output format**. This represents the format in which alarm content needs to be emailed. The default format is as follows:

```
<eG_Alarm>\n<Priority>$prior</Priority>\n<AlarmId>$alarmid</AlarmId>\n<User>$user<
/User>\n<ComponentName>$cname</ComponentName>\n<ComponentType>$ctype</ComponentTyp
e>\n<Layer>$layer</Layer>\n<Problem>$pdesc</Problem>\n<Service(s)>$Service</Servic
e(s)>\n<DD>$DD</DD>\n</eG_Alarm>
```

As stated earlier, a single TT mail can comprise of details pertaining to numerous alarms. Every such alarm definition within a TT mail will typically begin with the tag *<eG_Alarm>* and end with the tag *</eG_Alarm>*. This essentially indicates that the details contained within these tags pertain to a single alarm. These tags can be changed if so required. For example, you can specify *<Alarm_info>* and *</Alarm_Info>* instead of *<eG_Alarm>* and *</eG_Alarm>*. The variables **$prior**, **$alarmid**, **$user**, **$cname**, **$ctype**, **$layer**, **$pdesc**, **$service** , and **$dd**, during run-time, will display the alarm priority, alarm id, the user associated with the problem component, the problem component's name, the problem component type, the problem layer, the problem description, the service affected by the problem, and the detailed diagnosis of the problem (if any), respectively. In a TT mail, the value of each of the defined variables will be enclosed within the opening and closing tags defined in the **Mail output format**. For example, take the case of the specification *<Priority>$prior</Priority>*. In the TT mail for a *critical* alarm, this specification will appear as *<Priority> critical </Priority>*. These tags serve as qualifiers for the enclosed values. In other words, they indicate what value is displayed within. These tags can also be modified, if need be. However, the dollared variable names cannot be changed. '**/n**' acts as a separator for the values. A sample TT mail output has been provided below:

```
<eG_Alarm>
<Priority>Normal</Priority>
<AlarmId>2</AlarmId>
<User>john</User>
<ComponentName>king:7001</ComponentName>
<ComponentType>WebLogic_server</ComponentType>
<Layer>WL_SERVICE</Layer>
<Problem>Many invocations{DD}</Problem>
</eG_Alarm>
```

7. If the problem component is associated with multiple users, then the **$user** variable, if used in the **Mail output format**, will display a comma-separated user list. Typically, these users will be the ones responsible for resolving the issues with the corresponding component. However, some users will be authorized by the eG Enterprise system to oversee the performance of all the components in the environment (for eg., the default users of the eG Enterprise system – *admin* and *supermonitor*). In most cases, the responsibilities of such users will be more 'supervisory' in nature, and not administrative – i.e. might not involve troubleshooting and problem redressal.

Therefore, by default, the eG Enterprise system will hide the ID of this user from the user list displayed in the TT mail output. To ensure that the user list displays such a user ID too, set the **Include user details in mail** flag to **Yes**. By default, this parameter will be set to **No**.

8. Typically, every new alert generated by the eG Enterprise system will be associated with a unique alarm ID. By default, when every new alarm is processed by eG's trouble ticketing engine, a different alarm ID is generated by the engine, which will be of the format: *<ManagerIP>:<a long value>*. Instead of this TT engine-generated alarm ID, if you want the original, manager-generated alarm ID to be associated with the alarm and be part of the TT mail output, set the **Use unique ID** flag to **No**. By default, this flag is set to **Yes**.

9. If need be, you can make sure that the TT mails indicate the alarm priority using numbers instead of priority names such as *critical, major, minor,* or *normal*. For this purpose, you will have to set the **Priority as numeric** flag in the Figure 2.1 to **Yes**. By default, this flag is set to **No**, indicating that the priority of an alarm is indicated using the priority name by default. If this flag is set to **Yes** instead, then the default priority name-number mappings defined in the **[TMAIL]** section of the **eg_services.ini** file (in the **<EG_INSTALL_DIR>\manager\config** folder) will automatically apply. These default mappings are as follows:

```
Critical=1
Major=2
Minor=3
```

According to these mappings, if the **Priority as numeric** flag is set to **Yes**, then, in every TT mail sent subsequently, critical priority will be represented by number 1, major priority by number 2, and minor priority by number 3. If required, you can even change the numbers that should represent the alarm priorities. For instance, your priority name-number mapping can be as follows:

```
Critical=10
Major=11
Minor=12
```

10. Also, by setting the **Apply priority for all TT mails** flag to **Yes** or **No**, you can indicate whether the **AllowedAlarms** setting applies only to each new alarm ID that is raised by the eG manager or to modified alarms as well. As already stated, if an existing alarm has been modified, the eG manager retains the earlier assigned alarm ID for this alarm. The following are considered alarm modifications:

    - A change in the alarm priority: This could be a switch to a higher or lower priority.
    - A change in the alarm description: For example, originally, a usage-related alarm may have been raised on disk 'D' of a server. Later, disk 'C' of the same server might have experienced a space crunch, causing another alarm to be raised. In this case, the description of the original alarm will change to indicate that both disks C and D are experiencing a problem, but the alarm ID will not change. Changes in alarm description may also happen if additional tests being run for the same layer indicate a problem. A change may involve either an addition to the description (as in the example above) or a removal of one or more descriptors (e.g., the space usage of disk 'C' in the example / above returning to a normal condition).
    - A change in the list of impacted services

If the **Apply priority for all TT mails** flag is set to **Yes**, then the eG manager will send an email alert into the TT system even if one of the above modifications occur on an existing alarm, as long as the priority of the modified alarm belongs to the list of priorities configured against **Alarm preferences** in the **Common Settings** page (see Figure 1.1). On the other hand, if the **Apply priority for all TT mails** flag is set to **No**, then the eG manager will send email alerts only for every new alarm ID. In this case, the manager will ignore all subsequent changes to the priority of the alarm.

11. Finally, click the **Update** button.

You can even configure the specific tests for which TT mails are to be sent using the **TestsList** parameter in the **[TTMAIL]** section. By default, this parameter is set to *All*, indicating that the eG manager, by default, sends out TT mails for alarms related to all tests. To restrict TT mail transmission to specific tests, provide a comma-separated list of tests against **TestsList**. While providing test names here, make sure you provide the *<internaltestnames>* and not the display names. For instance, say, you want TT mails to be sent only when the eG manager raises alarms for the **Processes** test and the **System Details** test. To achieve this, your **TestsList** specification should be as follows:

```
TestsList=ProcessTest,SystemTest
```

In the specification above, the internal name for **Processes** test is *ProcessTest*, and the same for **SystemDetals** test is *SystemTest*. To determine the *internal name* of a test, do the following:

- Open the **eg_lang*.ini** file (from the **<EG_INSTALL_DIR>\manager\config** directory), where * is the language code that represents the language preference that you have set using the **USER PROFILE** page. In this file, the component types, measure names, test names, layer names, measure descriptions, and a wide range of other display information are expressed in a particular language, and are mapped to their eG equivalents.

- Now, search the **eg_lang*.ini** file for the test name of interest to you. For example, to know the internal name of the **SystemDetails** test, search the language file for the text, **SystemDetails**.

- Since the eG equivalent of the **SystemDetails** test is *SystemTest*, you will find a specification to that effect in the **[TEST_NAME_MAPPING]** section of that file. For our example, the specification would be as follows:

```
SystemTest=SystemDetails
```

3

Trouble Ticket Integration Using SNMP Traps

eG Enterprise is capable of transmitting alarms generated by the eG manager via SNMP traps to an SNMP management console such as HP OpenView, Netcool etc. In this case typically, for every alert that is generated in the eG Enterprise system, individual traps will be generated. In other words, a new SNMP trap will be sent out whenever a new problem is detected or an old problem changes (eg., a change in alarm priority, a change in alarm description, a change in the services impacted, etc.). This means that if a problem occurs across multiple descriptors of the same test, traps will be sent for each of the descriptors to the SNMP management system. Sometimes however, you may want to track an issue closely so that you can tell when it actually occurred and when it 'changed'. To ensure this, the eG Enterprise system provides you with the option to implement SNMP traps in the trouble ticketing integration module. When this is done, each trap sent by the eG manager to the third-party SNMP management console will be accompanied by a unique TTID. The SNMP console should be able to recognize the TT ID, and if a trap with the same TT ID re-appears, it should overwrite the last state it has for that TT ID. This way, the SNMP management system can differentiate between new alarms and modified alarms.

To enable integration via SNMP traps, the following steps should be followed:

- Configure at least one SNMP manager for the eG manager to integrate with. Refer to Section 3.1.1 to know how to configure an SNMP manager.
- Configure the eG manager to send traps to the SNMP manager(s) as trouble tickets (i.e., with a TTID). Refer to Section 3.1.2 to know how to configure the eG manager so.

3.1 How to Enable TT Integration over SNMP Traps?

3.1.1 Configuring a Third-party SNMP Manager

To configure the SNMP managers/trap receivers to which the eG manager needs to send SNMP traps, do the following:

1. Select the **Receivers and Settings** option from the **SNMP Traps** menu in the **Alerts** tile.
2. Figure 3.1 will then appear.

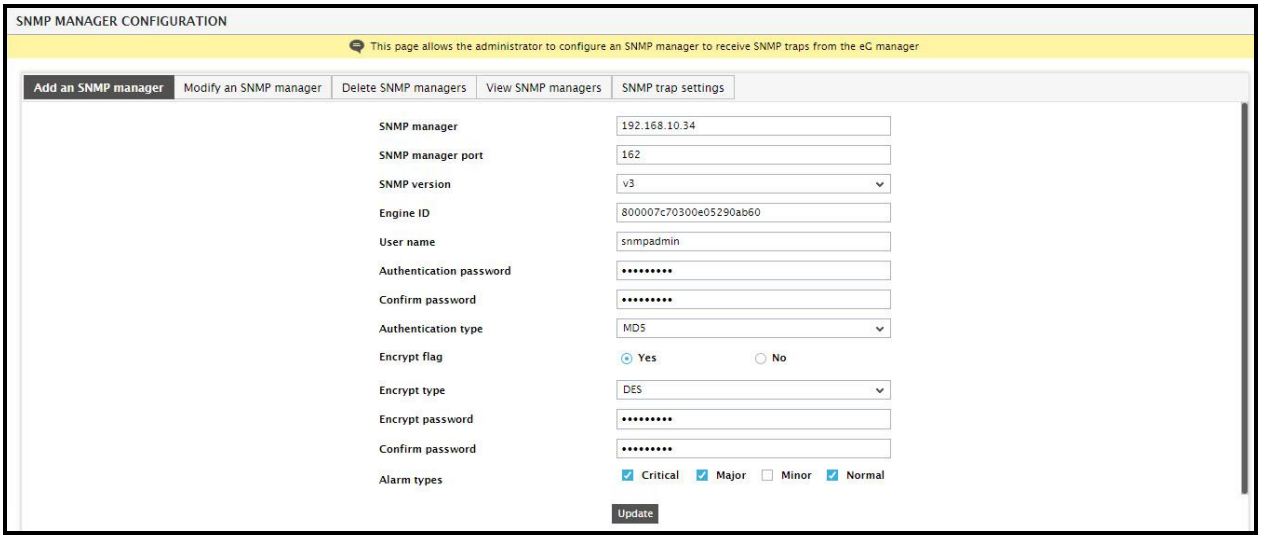

Figure 3.1: Adding an SNMP manager

3. The IP address of the SNMP manager on which the SNMP manager application is executing has to be provided in the **SNMP manager** text box in Figure 3.1. The port number on which the SNMP manager is listening for traps from the eG manager is to be specified in the **SNMP manager port** field. The default port is 162.

4. By default, the eG agent supports SNMP version 1. Accordingly, the default selection in the **SNMP version** list is **v1**. However, if a different SNMP framework is in use in your environment, say SNMP **v2** or **v3**, then select the corresponding option from this list.

5. The **SNMP community** field appears only if the **SNMP version** chosen is **1** or **2**. Here, specify the community string that is used by an eG manager to report alarm information via SNMP to an SNMP manager.

6. If the **SNMP version** is **3**, then you will have to specify the following parameters (see Figure 3.1):

   - **Engine ID**: Specify the engine ID of the trap sender. This should be in hexadecimal.
   - **User name**: As SNMPv3 traps require authentication, specify a valid user name here.
   - **Authentication password**: Enter the password of the above-mentioned **User name**.
   - **Confirm password**: Confirm the **Authentication Password** by retyping it here.
   - **Authentication type**: Choose the authentication algorithm using which SNMP v3 converts the specified **User name** and **Authentication password** into a 32-bit format to ensure security of SNMP transactions. You can choose between the following options:
     - **MD5** - Message Digest Algorithm
     - **SHA** - Secure Hash Algorithm
   - **Encrypt flag:** By default, the eG manager does not encrypt SNMP traps. Accordingly, this parameter is set to **No** by default. To ensure that SNMP traps sent by the eG manager are encrypted, select the **Yes** option.
   - **Encrypt type**: If the **Encrypt flag** is set to **Yes**, then you will have to mention the encryption type by selecting an option from the **Encrypt type** list. SNMP v3 supports the following encryption types:
     - **DES** - Data Encryption Standard
     - **AES** - Advanced Encryption Standard

- **Encrypt password:** Specify the encryption password here.
- **Confirm password**: Confirm the encryption password by retyping the password here.
- Select the required check boxes against **Alarm types** to indicate which alarm priorities need to be sent out as SNMP traps to the third-party SNMP management console.

7. Finally, click the **Update** button to add the new SNMP manager.
8. Next, proceed to configure the SNMP trap settings. For this, click the **SNMP trap settings** tab page in Figure 3.1. Figure 3.2 will appear.

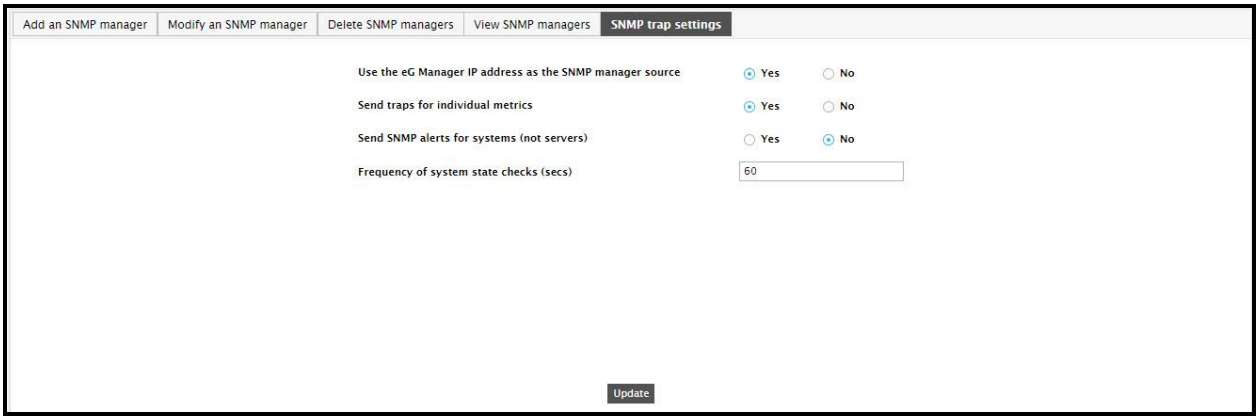

Figure 3.2: Configuring the SNMP trap settings

9. To ensure that the SNMP trap's source field includes the IP address of the eG manager from which the traps originated, set the **Use the manager's IP address (not name) in the SNMP trap's source field** flag in Figure 3.2 to **Yes**. Setting this flag to **No** will include the host name of the eG manager in the source field.
10. Set the **Send traps for individual metrics** flag to **Yes**, if you want the eG manager to send out SNMP traps whenever:
    - A new alarm is raised on a measure
    - An existing alarm related to a measure changes - an alarm change can be a change in the alarm priority, a change in the alarm description (eg., an addition/removal of a descriptor from an alarm), or change in the list of impacted services

    If you set the **Send traps for individual metrics** flag to **No**, then the test will send only new alarms raised on a measure as SNMP traps, and will disregard alarm changes. By default, this flag is set to **Yes**.
11. If multiple applications operate on a single host - i.e., if multiple components are managed using the same nick name - then, you can set the **Send SNMP traps for systems (not servers)** flag to **Yes**, so that the eG manager generates an SNMP trap for only the very first alarm that is raised on that nick name. In this case therefore, subsequent alarms for the same nick name will not be considered for trap generation. To turn off this capability, set the **Send SNMP traps for systems (not servers)** flag to **No**.
12. Indicate the frequency (in seconds) with which the eG manager needs to check the state of a host for SNMP trap generation, in the **Frequency of system state checks (secs)** text box. The default is 60 seconds (i.e., 1 minute).

13. Finally, click the **Update** button to enable the transmission of SNMP traps.

3.1.2 Sending Trouble Tickets over SNMP Traps

For this, do the following:

1. Select the **Manager** option from the **Settings** tile.
2. When Figure 3.3 appears, expand the **Trouble Ticket Integration** node in the **MANAGER SETTINGS** tree-structure in the left panel, and select the **Mail / SNMP** sub-node within.

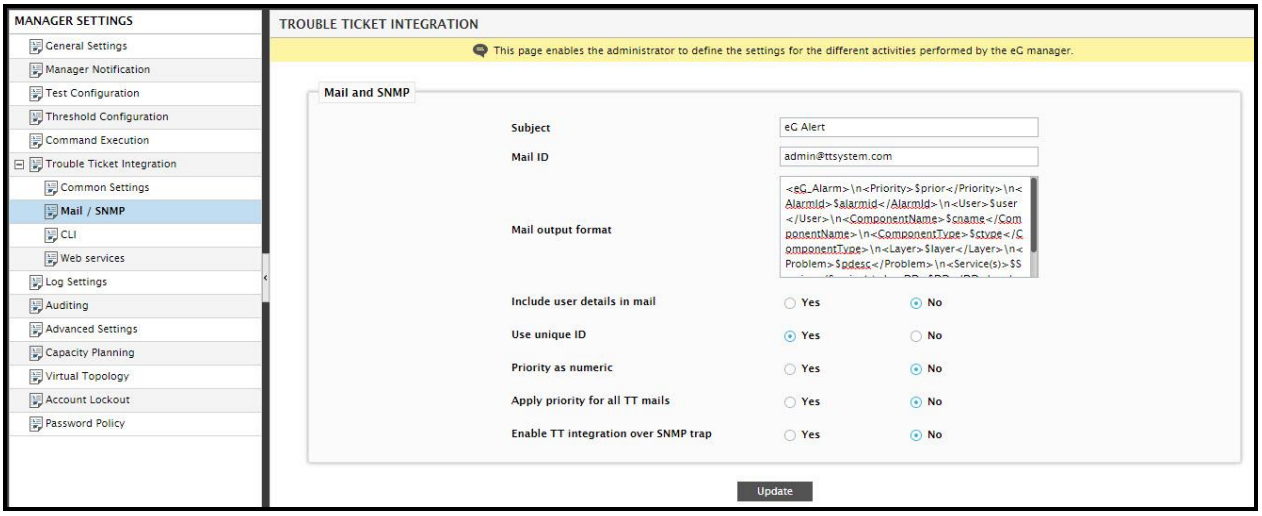

Figure 3.3: Enabling TT integration over SNMP traps

3. Finally, click the **Update** button.
4. Upon clicking **Update**, the **'trouble tickets as traps' capability will be automatically enabled for all the SNMP managers that have been pre-defined in the eG Enterprise system**.

Once the capability is enabled, then, for ech alert generated by the eG manager, an SNMP trap carrying the following information will be sent to all the third-party SNMP management systems registered with the eG Enterprise system:

- **TT ID** - unique identifier of the trouble ticket; TTID will be generated based on the settings defined for the third-party SNMP management system;
- **Component name** - The name of the problem component
- **Component type** - The problem component type (as it appears in the **Current Alarms** window of the eG monitoring console)
- **Layer** - The problematic layer
- **Problem Time** - the date/time when the problem started (as in the eG alarm window). The format to be used for date and time will be taken from the default setting of the eG Enterprise suite.
- **Problem description** - A brief description of the problem

Once an SNMP manager is configured, the eG manager will start sending SNMP traps to that manager, even if the **Enable TT integration over SNMP traps** flag is set to **No**. Such SNMP traps will be governed by the **Alarm types** and **SNMP Trap Settings** configured using Figure 3.1 and Figure 3.2, respectively. Moreover, such traps will not carry the TTID.

On the other hand, as soon as the **Enable TT integration over SNMP traps** flag is switched on, the eG manager will start sending SNMP traps with TTID to the same SNMP manager, alongside the SNMP traps without the TTID. The traps with TTID will be governed by the **Alarm preferences** and other settings configured in the **Common Settings** page of Figure 1.1. Also, such traps will not be affected by the **Alarm types** and **SNMP Trap Settings** configured using Figure 3.1 and Figure 3.2.

If you select a set of **Alarm types** to be sent as SNMP traps using the **SNMP MANAGER CONFIGURATION** page (see Figure 3.1), and also set **Alarm preferences** for TT integration in the **Common Settings** page of Figure 1.1, then, once the **Enable TT integration over SNMP traps** flag is set to **Yes**, the eG manager will send separate traps for the alarm priorities chosen from both pages. For instance, say that the *Critical* and *Major* check boxes are chosen from the **Alarm types** section of Figure 3.1 and only the *Critical* check box is selected from the **Alarm preferences** section of Figure 1.1. In this case, the eG manager will send two traps for every *Critical* alarm – one with TTID and one without TTID – and one trap for every *Major* alarm. The *Critical* alarm with the TTID will be sent from eG's trouble ticketing engine to match with the **Alarm preferences** setting for TT integration, and the *Critical* alarm without the TTID will be sent from eG's SNMP trap engine to match with the **Alarm types** setting.

3.2 Enabling Logging of SNMP Trap Transmissions

To know which alarms have been sent as traps to the SNMP manager, you can configure the automatic creation of log files, where the time at which traps were sent and the details of the traps are logged. To enable logging for the SNMP trap transmissions, do the following:

1. Edit the **eg_services.ini** file in the **<EG_INSTALL_DIR>\manager\config** directory.
2. In the **[MISC_ARGS]** section of that file, set the **SnmpTrapLogsEnabled** flag to **Yes** and and save the file. Once this is done, a log file named **<IP_address_of_SNMP_manager>_snmptrap_log** will be automatically created in the **<EG_INSTALL_DIR>\manager\logs** directory on the eG manager host.
3. By default, the eG manager will keep writing to the **<IP_address_of_SNMP_manager>_snmptrap_log** file, until the size of the log file grows upto 2 MB. As soon as the log file size crosses 2 MB, the manager will automatically copy the contents of the **<IP_address_of_SNMP_manager>_snmptrap_log** file to a new **<IP_address_of_SNMP_manager>_snmptrap_log.1** file, and will then continue writing newer logs to the original **<IP_address_of_SNMP_manager>_snmptrap_log**. This default behavior is governed by the **SnmpTrapLogMaxRollSize** parameter in the **eg_services.ini** file. By default, this parameter is set to **2**, which represents 2 MB. If you want the log file rotation to occur much sooner or later, then set the **SnmpTrapLogMaxRollSize** parameter accordingly. For instance, if you want the eG manager to wait until the size of the **<IP_address_of_SNMP_manager>_snmptrap_log** file becomes 4 MB to copy its contents to the **<IP_address_of_SNMP_manager>_snmptrap_log.1** file, then, set the **SnmpTrapLogMaxRollSize** to **4**, and save the file.
4. By default, the eG manager will continue to rotate the log files (as described in step 3 above) until a maximum of 10 log files are created. After this point, the last log file created (i.e.,

<IP_address_of_SNMP_manager>_snmptrap_log.9 file, by default), will be deleted, as it contains the oldest logs. This default behavior is governed by the **SnmpTrap_log_max_files** parameter in the **eg_services.ini** file. This parameter is set to **10** by default. If you want the roll over to occur much sooner or later, then change the value of this parameter accordingly. For instance, if you want the log files to roll over only if the total number of log files created crosses 20, then set the **SnmpTrap_log_max_files** parameter to **20**, and save the file.

4

Trouble Ticket Integration Using the eG TT CLI

The eG manager can also be configured so that whenever it detects a new alarm, a change in an existing alarm, or a closure of an existing alarm, it executes a command with the appropriate parameters indicating the current status of the alarm. **Note that this capability is available for stand-alone Windows managers, and Windows managers operating in redundant clusters only.** Prior to configuring this capability on such managers, the following pre-requisites need to be fulfilled:

To configure the command to be executed, do the following:

1. Select the **Manager** option from the **Settings** tile.
2. When Figure 4.1 appears, expand the **Trouble Ticket Integration** node in the tree-structure in the left panel of Figure 4.1, and click the **CLI** sub-node.
3. The right panel will then change to display a **CLI** section.

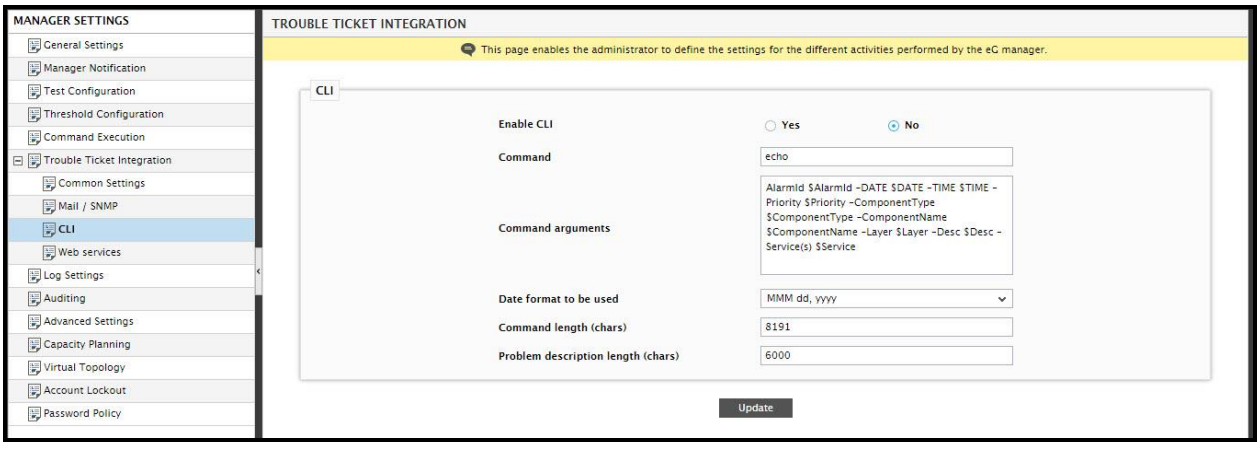

Figure 4.1: eG Manager settings for TT System CLI integration

4. Set the **Enable CLI** parameter in Figure 4.1 to **Yes** to enable this capability.
5. In the **Command** text box, **echo** is displayed by default, indicating that the eG manager will execute an *echo* command by default to communicate with the TT system.
6. The **Command Arguments** text box displays the default input parameters that the **echo** command takes during

execution. These default parameters are as follows:

```
AlarmId $AlarmId -DATE $DATE -TIME $TIME -Priority $Priority -ComponentType
$ComponentType -ComponentName $ComponentName -Layer $Layer -Desc $Desc -Service(s)
$Service
```

As you can see, the each parameter is represented by a qualifier and a variable name. While the qualifier is typically prefixed by a hyphen (-), the variable name is prefixed by a $ symbol. These variables will be substituted by actual values during runtime. Using the qualifiers, you will be able to tell what value follows. For instance, at runtime, the parameter ***–Priority $Priority*** could appear as ***–Priority Critical.*** This implies that the **Priority** of the problem is **Critical**.

- You can alter the qualifier if you need to, but the variable names (the $ preceded strings in the previous example) should not be changed.
- The position of the qualifier-variable pairs can be changed in the command line – for instance, you can move the ***–Service $Service*** parameter to appear next to the ***AlarmID $AlarmId*** if you want to.
- Though the eG manager executes a command line by default, you can change this **Command** specification to execute a batch file/script file/executable instead.
- The **Command Arguments** specification can include any special character that will work from the Windows command prompt or Unix shell, except the # (hash) character, which is used as a separator between the command and the arguments.
- On Windows, the eG manager runs as a service, and hence, has access to all the sysem-defined environment variables. You can use these variables as required in the script that is invoked by the **Command** specification.

Given below is the list of parameters the default **Command Arguments** display takes, and a brief description of each parameter:

- ***AlarmId $AlarmId*** – unique identifier of the alarm
- ***-DATE $DATE*** – the date on which the problem occurred
- ***-TIME $TIME –*** the time at which the problem occurred.
- ***-Priority $Priority*** – the problem priority - whether Critical, Major, Minor, Normal
- *-ComponentType $ComponentType* – The problem component type
- *-ComponentName $ComponentName* – The problem component name
- ***-Layer $Layer*** – the protocol layer to which the problem relates.
- ***-Desc $Desc*** – a brief description of the problem. This will include a pipe (|) separated list of the following fields – the site name (if relevant to the test), test name, the alarm string (a textual description of the problem), and measurement host. The description is not applicable if the alarm severity is **Normal**.
- ***–Service(s) $Service*** – If more than one service is impacted, this will include a comma-separated list of services.

- **-DD $DD** - **This parameter is not available by default**. If required, you can configure this parameter additionally for the command so that, the output includes detailed diagnosis information. In the output, ***$DD*** will be represented in the following format:

  "DDcolumn1 DDcolumn2 DDcolumn3 …~#~DDdata1~!~DDdata2~!~DDdata3~!~..."

  In the eG user interface, the detailed diagnosis pertaining to a single test/measure/descriptor combination is typically presented in a tabular format, with rows and columns. Accordingly, in the command output for a specific test-measure-descriptor combination, the variables *DDcolum1, DDcolumn2*, etc., will be substituted by the names of the columns in the detailed diagnosis, and the variables *DDdata, DDdata2, etc.,* will report the values that correspond to each column.

- The detailed diagnosis information reported in the CLI output will typically be the DD information available in the eG database, at the time of command execution by the eG manager.
- A single alarm could have multiple tests, measures, and descriptors associated with it. For each test-measure-descriptor combination, there will be a corresponding DD entry in the command output line (subject to the length restriction discussed in the **Limitations** section below). The DD output for a single test-measure-descriptor combination will include the column names and data. In the output, these columns and their corresponding data will be separated using the separator ~#~. If a specific combination does not have DD configured or there is no DD reported for that descriptor, a value "-" will be reported.
- The DD output for each test-measure-descriptor combination will be separated using #~#.
- Detailed diagnosis information for a test-measure-descriptor combination could include rows of data. In the command output, the separator !~! is used to separate multiple rows of DD data.
- Each row of data in the DD would report values for several columns. The values that correspond to each column will be separated using ~!~ in the DD output.

A sample standard output of the default **Command** specification is given below:

```
AlarmId "192.168.10.133_1264839507765" -DATE "Jan 30, 2010" -TIME "13:48:24" -
Priority "1" -ComponentType "Host system" -ComponentName "win: NULL" -Layer
"Operating System" -Desc "-|SystemDetails|High CPU
utilization{Processor_0}|win,-|SystemDetails|Free memory is
low{Processor_0}|win" -Service "-" -DD " PID %CPU ARGS
~#~692~!~1.60~!~csrss!~!760~!~0.53~!~services!~!6960.53~!~ js #~# PID %MEM
ARGS~#~4032~!~7.83~!~tomcat!~!2112~!~6.33~!~firefox!~!3472~!~5.52~!~dbvis"
```

According to the above output, the following alarm information will be sent to the third-party TT system:

Qualifier	Variable
AlarmId	192.168.10.133_1256878963888
DATE	30/10/2009
TIME	10:32:43
Priority	Critical
ComponentType	Generic
ComponentName	gen133:NULL
Layer	Application Processes
Desc	In our example, the command line output clubs the information pertaining to two alarms related to a single test. The description of the first alarm indicates the following: ▪ the site affected (if applicable) : In the case of our example, no web site has been impacted by the Critical problem; therefore, only a '-' (hyphen) is displayed instead in the output line. ▪ The test that reported the problem: In the case of our example, this is **SystemDetails** test ▪ The alarm description: In the case of the sample output, this is - **High CPU utilization { Processor_0}** ▪ The measurement host: In the case of the sample output, this is – **win** The description of the second alarm includes the following: ▪ the site affected (if applicable) : In the case of the second alarm also, no web site has been impacted by the Critical problem; therefore, only a '-' (hyphen) is displayed instead in the output line. ▪ The test that reported the problem: In the case of our example, this is **SystemDetails** test ▪ The alarm description: In the case of the sample output, this is - **Free memory is low{Processor_0}** ▪ The measurement host: In the case of the sample output, this is – **win**
Service	Since no service has been impacted by the problem at hand, only a '-' is displayed here instead

DD	The sample output above includes the DD information pertaining to two test-measure-descriptor combinations. For the first test-measure-descriptor combination - i.e., for the **CPU utilization** measure of the descriptor **Processor_0** reported by the **SystemDetails** test - the DD output includes the following: ▪ The columns in the DD are as follows: **PID**, **%CPU**, and **ARGS** (as in 'arguments'). ▪ The DD for this test-measure-combination includes three rows of data. In the first row of data, the following values will be reported: ▪ **692** as the **PID** ▪ **1.60** as the **CPU%** ▪ **csrss** as the **ARGS** ▪ In the second row of data, the following values will be reported: ▪ **760** as the **PID** ▪ **0.53** as the **CPU%** ▪ **services** as the **ARGS** ▪ In the third row of data, the following values will be reported: ▪ **696** as the **PID** ▪ **0.53** as the **CPU%** ▪ **js** as the **ARGS** For the second test-measure-descriptor combination - i.e., for the **Free memory** measure of the descriptor **Processor_0** reported by the **SystemDetails** test - the DD output includes the following: ▪ The columns in the DD are as follows: PID, %MEM, and ARGS (as in 'arguments'). ▪ The DD for this test-measure-combination includes three rows of data. In the first row of data, the following values will be reported: ▪ **4032** as the **PID** ▪ **7.83** as the **MEM%** ▪ **tomcat** as the **ARGS** ▪ In the second row of data, the following values will be reported: ▪ **2112** as the **PID** ▪ **6.33** as the **MEM%** ▪ **firefox** as the **ARGS** ▪ In the third row of data, the following values will be reported: ▪ **3472** as the **PID** ▪ **5.52** as the **MEM%** ▪ **dbvis** as the **ARGS**

- DD information will be formatted for better display in eG user interface. From the CLI however, no such special formatting can be effected on the DD output.
- If an alert is raised on multiple descriptors, hyphen (-) will be provided if DD is not available for a specific descriptor.
- Detailed diagnosis could have special characters (e.g., double quotes) that may require special handling when passed to a CLI.
- If the command line execution fails when including DD, CLI will resume execution by excluding the DD, so that the alert does not get lost.

7. From the **Date format to be used** list box, select the format in which the date/time of the problem should be reported in the command output.
8. Specify the maximum permissible length of the command in the **Command length** text box. By default, the command line can have a maximum of 8191 characters. You can alter this default setting by specifying a length of your choice in the **Command length** text box. If the actual command length exceeds the specified limit, then the output will not return the list of affected services and the detailed diagnosis information; instead, an empty string will appear next to the ***–Services*** qualifier. If the command length continues to exceed the specified limit even after truncating the services list and the DD, the command execution will return an error.
9. Specify the length of the problem description in the **Problem description length** text box. If the actual problem description exceeds the specified length, the characters that fall beyond the specified limit will be truncated.
10. Finally, click on the **Update** button to save the changes.

As described above, the eG manager offers a high degree of flexibility in the configuration of the command that should be executed. The periodicity at which the eG manager checks alarms and determines what information it should send to a TT system is determined by the setting of the **MailCheckPeriod** attribute in the **[MISC_ARGS]** section of the **eg_services.ini** file (in the **<EG_INSTALL_DIR>\manager\config** directory). By default, this value is 180 seconds (3 minutes).

5

Trouble Ticket Integration Using a Web Services Framework

eG Enterprise can integrate with TT systems that support a web services framework. The eG manager establishes an HTTP/S connection to a web services URL on the third-party system and communicates alarm information to that TT system using its web services API. Upon receipt of an alarm, the TT system automatically generates/modifies/closes trouble tickets.

Without the need for any complex instrumentation, eG Enterprise can readily integrate with the following help desk systems via their web services interface:

- Manage Engine's ServiceDesk
- Autotask
- ServiceNow
- Remedy Force
- PagerDuty

Sections 5.1 to 5.5 that follow will discuss how eG integrates with each of the TT systems mentioned above.

To enable you to integrate with any other help desk system that supports a web services interface, eG Enterprise provides a proprietary web services-based integration framework, which can be easily fine-tuned to enable the integration. Section 5.5 of this chapter describes how this can be achieved.

5.1 Integrating with ManageEngine's ServiceDesk

ManageEngine ServiceDesk is a comprehensive Help Desk and Asset Management software that provides help desk agents and IT managers an integrated console to monitor and maintain the assets and IT requests generated from the users of the IT resources in an organization.

To integrate the eG manager with ServiceDesk, do the following:

1. Login to the eG administrative interface as *admin* with password *admin*.
2. Invoke the **Admin** tile menu and select the **Manager** option from the **Settings** tile.
3. In the **MANAGER SETTINGS** tree-structure in the left panel of Figure 5.1, expand the **Trouble Ticket Integration** node.

4. Next, select the **Web services** sub-node within.

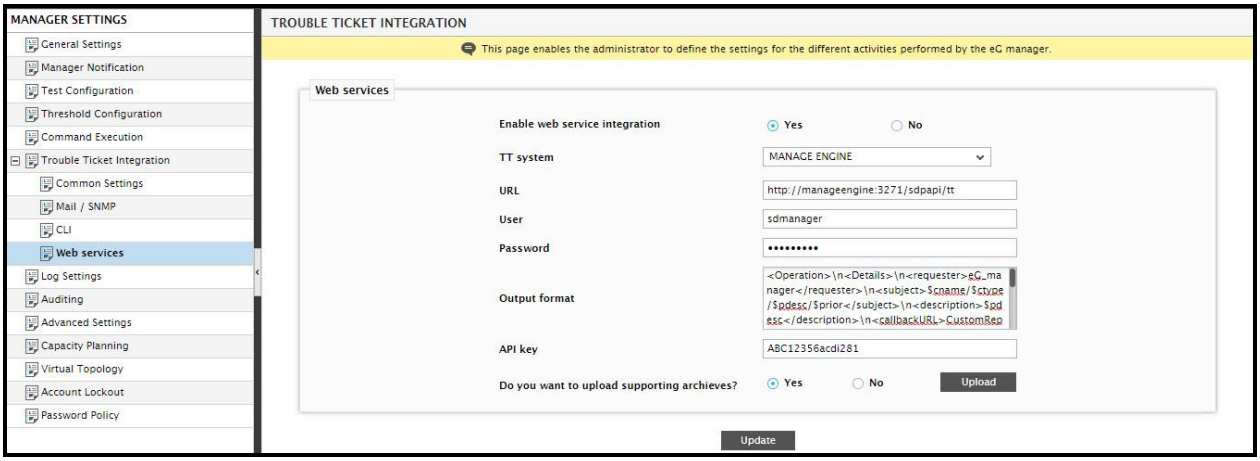

Figure 5.1: Integrating the eG manager with ManageEngine's ServiceDesk

5. To enable integration via the web services interface of ServiceDesk, set the **Enable web service integration** flag in Figure 5.1 to **Yes**. By default, this is set to **No**.

6. From the **TT system** drop-down, select **MANAGE ENGINE** as the TT system with which the eG manager should integrate.

7. Then, against the **URL**, specify the Web Services Description Language (WSDL) URL via which the eG manager should connect to ServiceDesk's web services interface.

8. If the connection needs to be authenticated, then provide a valid user name and password against **User** and **Password** text boxes, respectively.

9. ServiceDesk's web services API, known as REST API, is capable of interpreting problem inputs it receives and automatically generating/updating trouble tickets, only if the inputs are in XML format. This is why, by default, the eG manager sends its alarm output in XML format to ServiceDesk. The standard format, as displayed against O**utput format** in Figure 5.1, is as follows:

```
<Operation>\n<Details>\n<requester>eG_manager</requester>\n<subject>$cname/$ctype/
$pdesc/$prior</subject>\n<description>$pdesc</description>\n<callbackURL>CustomRep
ortHandler.do</callbackURL>\n<requesttemplate>-
</requesttemplate>\n<priority>$prior</priority>\n<layer>$layer</layer>\n<group>$us
er</group>\n<technician>eG_manager</technician>\n<level>1</level>\n<status>$status
</status>\n<service>$Service</service>\n</Details>\n</Operation>
```

The text enclosed within angular brackets – eg., *<Operation>* - are the XML tags that ServiceDesk's web services API recognizes. **These tags cannot be changed**. The text enclosed within an opening and a closing tag can either be static text or a variable. These tags and the values they contain are discussed hereunder:

<Operation>|n<Details>|n	The operations performed with REST API are based on the **'operation'** parameter and is sent to the url via HTTP POST method.
<requester></requester>	Should contain a static text indicating the user requesting for a trouble ticket. Using ServiceDesk's self-service portal, you can add, edit, or remove requesters. In the case of the standard output format above, a requester named *eG_manager* has apparently being created. Therefore, the static text *eG_manager* is enclosed within these tags. If the name of the requester is changed in ServiceDesk, make sure

	that this text is also changed.
<subject></subject>	Should contain a slash-separated list of variables representing the subject of the trouble ticket. In the case of the standard output format above, the following variables have been used in the subject and this is what they denote: ▪ $cname – at run time, this variable will change to display the exact name of the problem component ▪ $cname – at run time, this variable will change to report the problem component-type ▪ $pdesc- at run time, this variable will change to report the problem description. ▪ $prior – at run time, this variable will change to report the problem priority You can add / remove the subject variables at will, but you cannot **change the variable representation**. In other words, **you cannot change $cname to #name** when defining the output format.
<description></description>	Should contain the variable that represents the problem description – i.e., **$pdesc**. At run time, this variable will change to report the precise problem description.
<callbackURL</callbackURL>	Provide a valid callback URL within these tags. When the cause for an eG alarm is resolved, ServiceDesk will invoke this URL. The URL functions as a notification to the eG manager indicating that the ticket is resolved. If this URL (callback URL) is not provided, ServiceDesk will not perform any additional operation on the ticket.
<requesttemplate></requesttemplate>	Within these tags, specify the name of the request template to be used (if any). If no request template applies, as in the case of our example above, then enclose a – (hyphen) within these tags. Using ServiceDesk, one can create different incident/request templates, each configured with a set of fields using which an incident is to be reported.
<priority></priority>	Should contain the variable that represents the problem priority – i.e., **$prior**. At run time, this variable will change to report the precise problem priority.
<layer</layer>	Should contain the variable that represents the problem layer – i.e., **$layer**. At run time, this variable will change to report the problematic layer.
<group></group>	Should contain the variable that represents the user/support group to which the technician responsible for resolving this alarm belongs. At run time, this variable will change to report the exact user group.
<technician></technician>	Should contain the static text that indicates the name of the technician responsible for resolving the alarm. In the case of our standard output format, *eG_manager* is the technician. You can change this to reflect the name of any other technician who has been configured in ServiceDesk. .

<level></level>	Should indicate the support level. In the case of our standard output format, the support level is *1*.
<status></status>	Should contain the variable **$status** that represents the status of the ticket. At run time, this variable will change to report the exact status.
<service></service>	Should contain the variable **$service** that represents the name of the business service impacted by the problem. At run time, this variable will change to to report the correct business service name.

10. Then, specify the **API key** of the TT system. The eG manager will be able to communicate with ServiceDesk through this API key only. ManageEngine generates a unique key for the user, whose credentials are specified against **User** and **Password** parameters in Figure 5.1 above. **The eG manager will not be able to use the ServiceDesk API, if this key is not specified or an invalid key is specified**.

11. The implementation of eG integration with ServiceDesk exists in a Java class file named **ServiceDeskPlusHelper.java**. Additionally, the API files and library files that support this class file are jarred in **ServiceDesk.jar** file. In eG, this jar file is referred to as the **TT integration archive file**. By default, the implementation class file and the jar file are bundled with the eG manager and are available in the **<EG_INSTALL_DIR>\manager\lib** directory on the eG manager host. However, if later, changes are made to these files, you will have to re-upload them to the eG manager. In such a case, set the **Do you want to upload the supporting archives?** flag to **Yes**, and click the **Upload** button alongside. This will invoke Figure 5.2.

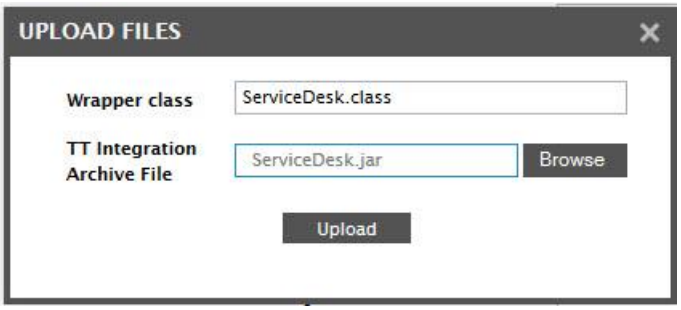

Figure 5.2: Uploading the TT integration jar file to the eG manager

Here, specify the full path to the **Wrapper class** file – i.e., the **ServiceDeskPlusHelper.java** file - and the **TT Integration Archive File** – i.e., the **ServiceDesk.jar** file - to be uploaded. Use the **Browse** button for the path specification. Finally, click the **Upload** button.

12. Finally, click the **Update** button.

5.2 Integrating with ServiceNow

ServiceNow is a platform-as-a-service (PaaS) provider of Service Management (SM) software for the entire enterprise. The ServiceNow platform also supports the incident management process with the ability to log incidents, classify according to impact and urgency, assign to appropriate groups, escalate, and manage through to resolution and reporting.

To integrate the eG manager with the incident management process of ServiceNow, do the following:

1. Login to the eG administrative interface as *admin* with password *admin*.

2. Invoke the **Admin** tile menu and select the **Manager** option from the **Settings** tile.

3. In the **MANAGER SETTINGS** tree-structure in the left panel of Figure 5.1, expand the **Trouble Ticket Integration** node.
4. Next, select the **Web services** sub-node within.

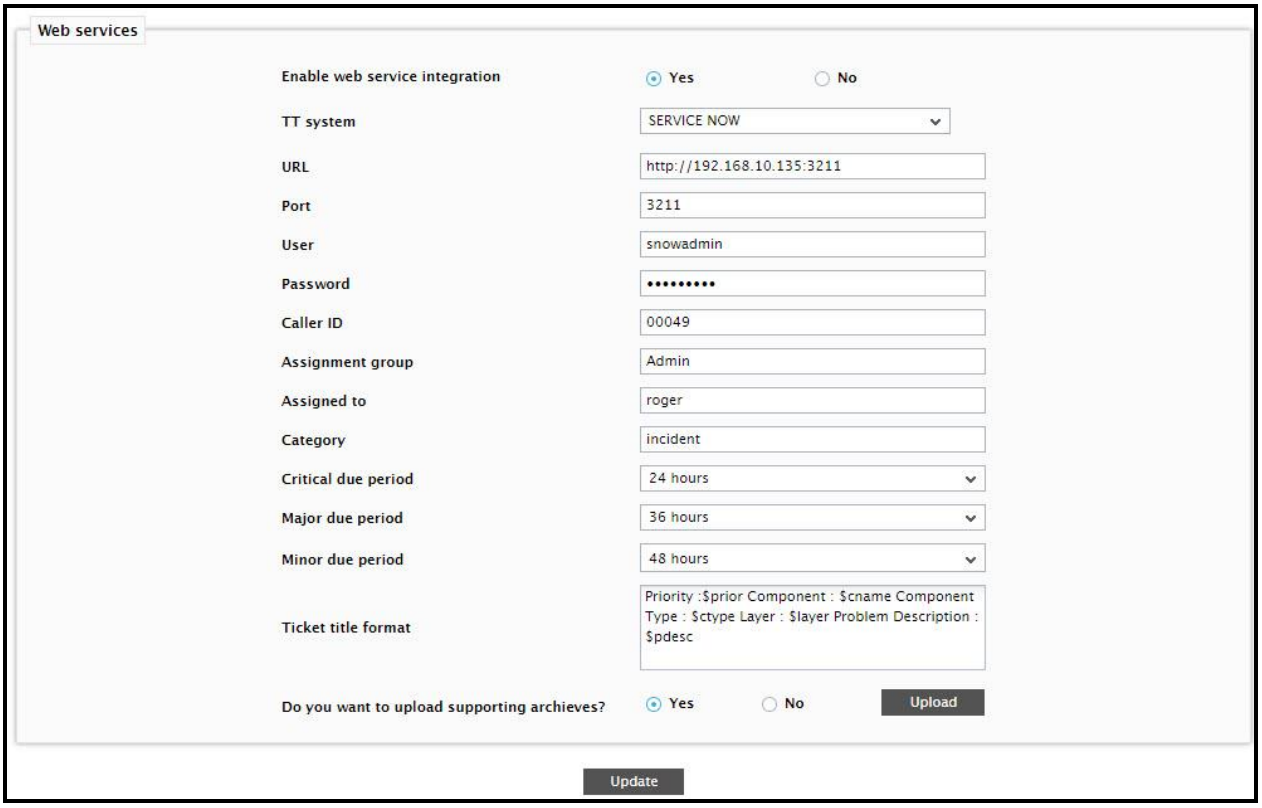

Figure 5.3: Integrating the eG manager with ServiceNow

5. To enable integration via the web services interface of ServiceDesk, set the **Enable web service integration** flag in Figure 5.3 to **Yes**. By default, this is set to **No**.
6. From the **TT system** drop-down, select **SERVICE NOW** as the TT system with which the eG manager should integrate.
7. Then, to enable the eG manager to connect to the ServiceNow installation in your environment, specify the following:
   - **URL**: The URL using which the eG manager should connect to the ServiceNow installation in your environment.
   - **Port:** The **Port** at which ServiceNow listens for problem information sent by the eG manager.
   - **User** and **Password**: The credentials of a user who has rights to access ServiceNow
8. Then, indicate the following as well:

- **Caller ID**: The ID of the user to whom the problems sent by the eG manager are to be assigned.
- **Assignment group**: The assignment group to which the eG alarms are to be assigned.
- **Assigned to**: The name of the user to whom the problems sent by the eG manager are to be assigned.
- **Category**: Indicate how to categorize the eG alarms in the ServiceNow system – whether as a request, enhancement, or an incident.
- **Critical due period**: The time (in millisecs) by which tickets of a **Critical** priority will be resolved.
- **Major due period**: The time (in millisecs) by which tickets of a **Major** priority will be resolved.
- **Minor due period**: The time (in millisecs) by which tickets of a **Minor** priority will be resolved.

9. Next, specify the format in which the title of the trouble ticket is to be displayed in the ServiceNow console. The default format is as follows:

```
Priority :$prior Component : $cname Component Type : $ctype Layer : $layer Problem Description : $pdesc
```

The text preceding the ':' (colon) in the format above indicates what information follows. The 'dollared' ($) text that follows the ':' (colon) is a variable, the value of which varies at run time, depending upon the information contained in the eG alarms. For example, in the default format above, *Priority* is a label that indicates that the information that follows the ':' is the priority of the alarm. The variable *$prior* that succeeds the ':' represents the alarm priority, and changes according to the priority of the actual alarm that is sent by the eG manager to ServiceNow. **While you can change the labels, you are advised against changing any of the variable names.**

The other variables that are part of the default format are discussed in the table below:

$cname	Will display the name of the problem component
$ctype	Will display the component type to which the problem component belongs
$layer	Will display the layer affected by the problem
$pdesc	Will display a brief problem description

So, if a *Critical* alarm raised by the eG manager for a high CPU usage problem detected in the *Operating System* layer of the Windows server, 192.168.10.15, is routed to ServiceNow, the web services API of ServiceNow will convert the alarm into a trouble ticket titled (by default) as follows:

```
Priority:Critical Component:192.168.10.15 Component Type:Windows Layer:Operating System Problem Description:High CPU usage
```

10. The implementation of eG integration with ServiceNow exists in a Java class file named **ServiceNowIntegrator.java**. Additionally, the API files and library files that support this class file are jarred in **HttpClient.jar** file. In eG, this jar file is referred to as the **TT integration archive file**. By default, the implementation class file and the jar file are bundled with the eG manager and are available in the **<EG_INSTALL_DIR>\manager\lib** directory on the eG manager host. However, if later, changes are made to these files, you will have to re-upload them to the eG manager. In such a case, set the **Do you want to upload the supporting archives?** flag to **Yes**, and click the **Upload** button alongside. This will invoke Figure 5.4.

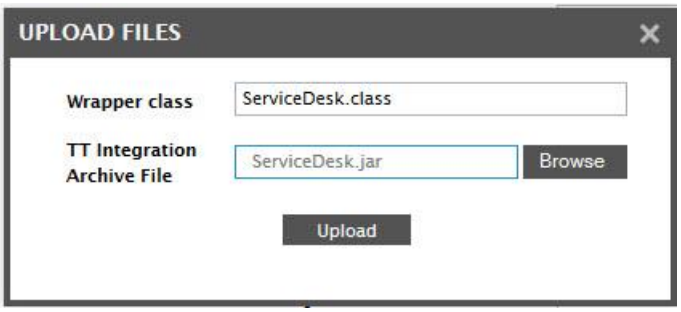

Figure 5.4: Uploading the TT integration jar file to the eG manager

Here, specify the full path to the **Wrapper class** file – i.e., the **ServiceNowIntegrator.java** file - and the **TT Integration Archive File** – i.e., the **HttpClient.jar** file - to be uploaded. Use the **Browse** button for the path specification. Finally, click the **Upload** button in Figure 5.4.

11. Finally, click the **Update** button in Figure 5.3.

5.3 Integrating with Autotask

Autotask provides a complete IT business management solution that combines Service Desk, CRM, Projects, Time & Expense, Billing and more.

To ensure that the eG manager integrates with the Service Desk module of Autotask via its web services API, do the following:

1. Login to the eG administrative interface as *admin* with password *admin*.
2. Invoke the **Admin** tile menu and select the **Manager** option from the **Settings** tile.
3. In the **MANAGER SETTINGS** tree-structure in the left panel, expand the **Trouble Ticket Integration** node.
4. Next, select the **Web services** sub-node within. The right panel will change as shown by Figure 5.5 below.

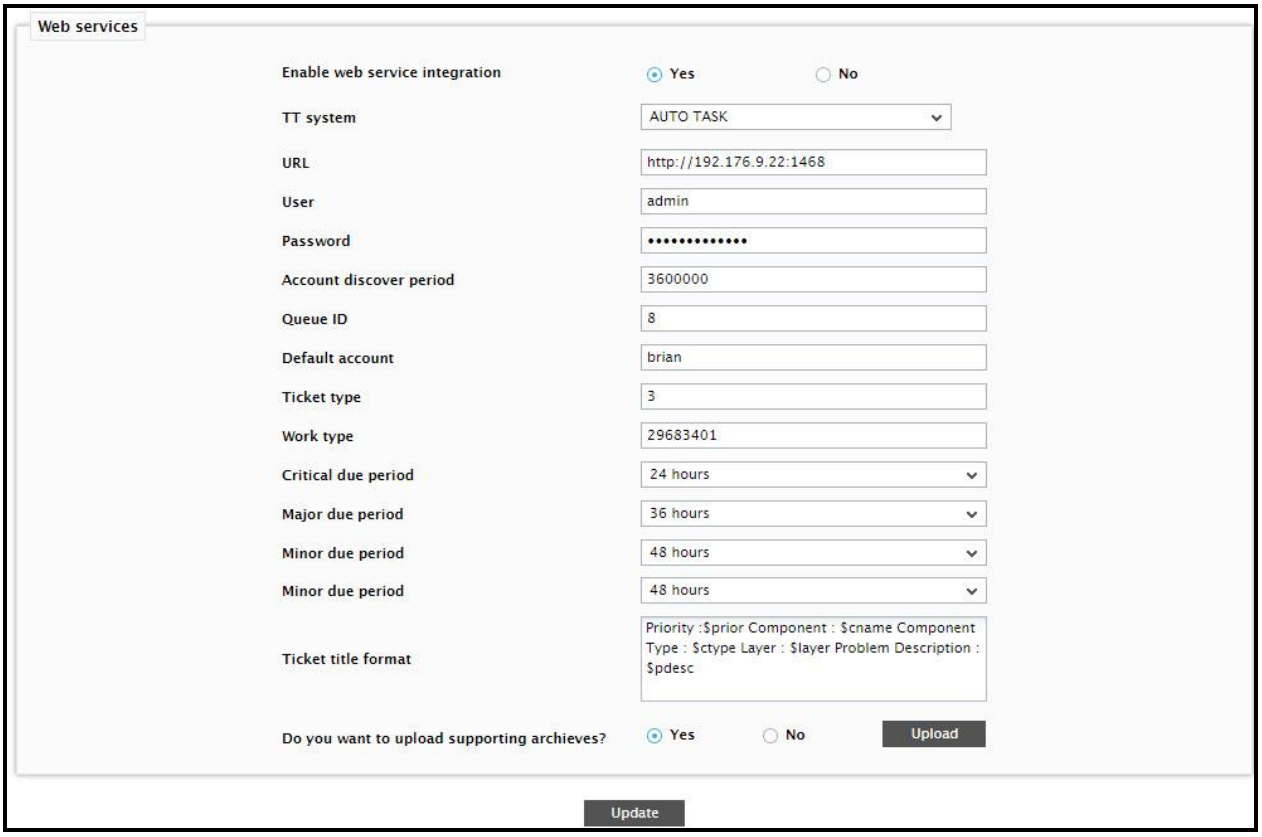

Figure 5.5: Integrating the eG manager with Autotask

5. To enable integration via the web services interface of Autotask, set the **Enable web service integration** flag in Figure 5.5 to **Yes**. By default, this is set to **No**.
6. From the **TT system** drop-down, select **AUTO TASK** as the TT system with which the eG manager should integrate.
7. Then, to enable the eG manager to connect to the Autotask installation in your environment, specify the following:
   - **URL:** The URL using which the eG manager should connect to Autotask.
   - **User** and **Password**: The credentials of a valid user who has rights to access Autotask.
8. Then, specify the following:
   - **Account discover period**: Enter how frequently the eG manager needs to auto-discover the user accounts configured in Autotask. By default, this is set to *3600000* millisecs. If an auto-discovered user account is mapped to one/more zones configured in eG, then Autotask will automatically assign the trouble related to the servers in those zones, to the corresponding user account. To map an auto-discovered user account with one/more zones, do the following:
     - Open the **eg_services.ini** file in the **<EG_INSTALL_DIR>\manager\config** directory, using an editor.
     - Typically, when user accounts are auto-discovered for the first time from Autotask, the eG manager automatically creates a section named **[AUTOTASK_ACCOUNT_MAPPING]** in the **eg_services.ini** file. All auto-discovered user accounts are inserted as entries in this section, in the following format:

[AUTOTASK_ACCOUNT_MAPPING]

<AccountName1>$@$<AccountNumber1>=
<AccountName2>$@$<AccountNumber2>=
. . .
. . .
. . .
<AccountNameN>$@$<AccountNumberN>=

To map a user account to one/more zones, just configure a comma-separated list of zones against the corresponding *<AccountName1>$@$<AccountNumber1>* entry in the **[AUTOTASK_ACCOUNT_MAPPING]** section, as shown below

[AUTOTASK_ACCOUNT_MAPPING]

<AccountName1>$@$<AccountNumber1>=east_coast_zone,zone_singapore,zone_london

- Finally, save the **eg_services.ini** file.

- **Queue ID**: Specify the ID of the queue to which eG alarms are to be assigned. By default, *8* is the queue to which all eG alarms are assigned.
- **Default account**: Indicate to which user account eG alarms are to be assigned by default. If a server on which eG has raised an alarm does not belong to any zone or any zone that is mapped to a user account, then such an alarm will be automatically assigned to the default user account. Likewise, if auto-discovered user accounts are not mapped to any zones configured in eG, then all eG alarms will be automatically assigned to the default user account.
- **Ticket type**: Indicate the type of ticket that is to be raised in Autotask for every eG alarm that the eG manager sends to it. By default, this parameter is set to *3*, which indicates that for eG alarms tickets of type, *Problem*, are generated by default in Autotask. You can change the value of this parameter to *1* to indicate *Service request* or *2* to indicate *Incident*.
- **Work type**: The work type to be assigned to the trouble tickets generated for the eG alarms. By default, this is *29683401.*
- **Critical due period**: The time (in millisecs) by which tickets of a **Critical** priority will be resolved.
- **Major due period**: The time (in millisecs) by which tickets of a **Major** priority will be resolved.
- **Minor due period**: The time (in millisecs) by which tickets of a **Minor** priority will be resolved.

9. Next, specify the format in which the title of the trouble ticket is to be displayed in the Autotask console. The default format is as follows:

```
Priority :$prior Component : $cname Component Type : $ctype Layer : $layer Problem
Description : $pdesc
```

The text preceding the ':' (colon) in the format above indicates what information follows. The 'dollared' ($) text that follows the ':' (colon) is a variable, the value of which varies at run time, depending upon the eG alarms. For example, in the default format above, *Priority* is a label that indicates that the information that follows the ':' is the priority of the alarm. The variable *$prior* that succeeds the ':' represents the alarm priority, and changes according to the priority of the actual alarm that is sent by the eG manager to Autotask. **While you can change the labels, you are advised against changing any of the variable names**.

The other variables that are part of the default format are discussed in the table below:

$cname	Will display the name of the problem component
$ctype	Will display the component type to which the problem component belongs
$layer	Will display the layer affected by the problem
$pdesc	Will display a brief problem description

So, if a *Critical* alarm raised by the eG manager for a high CPU usage problem detected in the *Operating System* layer of the Windows server, 192.168.10.15, is routed to Autotask, the web services API of Autotask will convert the alarm into a trouble ticket titled (by default) as follows:

```
Priority:Critical Component:192.168.10.15 Component Type:Windows Layer:Operating
System Problem Description:High CPU usage
```

10. The implementation of eG integration with Autotask exists in a Java class file named **AutoTaskIntegrator.java**. Additionally, the API files and library files that support this class file are jarred in **AutoTask.jar** file. In eG, this jar file is referred to as the **TT integration archive file**. By default, the implementation class file and the jar file are bundled with the eG manager and are available in the **<EG_INSTALL_DIR>\manager\lib** directory on the eG manager host. However, if later, changes are made to these files, you will have to re-upload them to the eG manager. In such a case, set the **Do you want to upload the supporting archives?** flag to **Yes**, and click the **Upload** button alongside. This will invoke Figure 5.6.

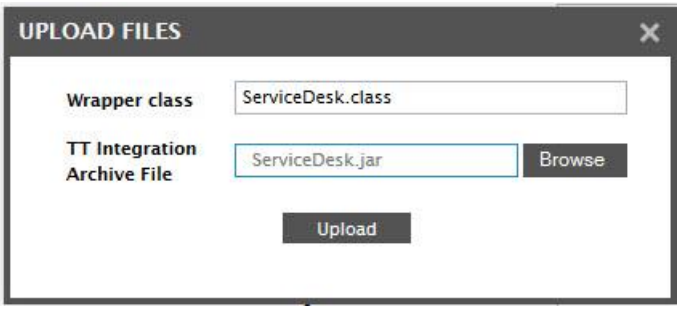

Figure 5.6: Uploading the TT integration jar file to the eG manager

Here, specify the full path to the **Wrapper class** file – i.e., the **AutoTaskIntegrator.java** file - and the **TT Integration Archive File** – i.e., the **AutoTask.jar** file - to be uploaded. Use the **Browse** button for the path specification. Finally, click the **Upload** button in Figure 5.6.

11. Finally, click the **Update** button in Figure 5.5.

5.4 Integrating with BMC RemedyForce

RemedyForce helps you streamline IT assistance and deliver it through a cloud-based, social platform.

To integrate the eG manager with BMC RemedyForce, do the following:

1. Login to the eG administrative interface as *admin* with password *admin*.

2. Invoke the **Admin** tile menu and select the **Manager** option from the **Settings** tile.
3. In the **MANAGER SETTINGS** tree-structure in the left panel, expand the **Trouble Ticket Integration** node.
4. Next, select the **Web services** sub-node within. The right panel will change as shown by Figure 5.7 below.

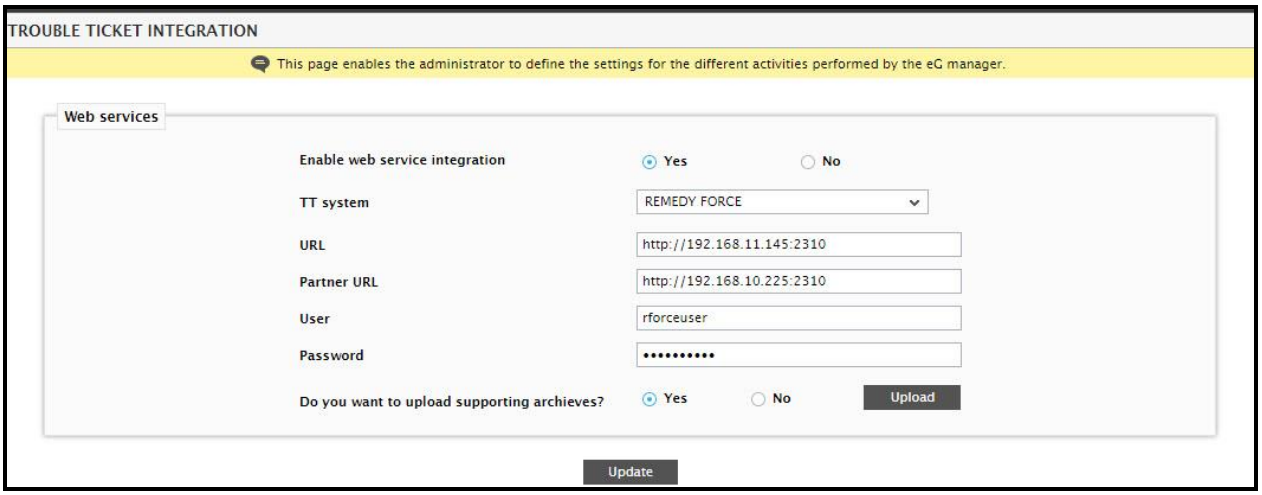

Figure 5.7: Integrating the eG manager with BMC RemedyForce

5. To enable integration via the web services interface of RemedyForce, set the **Enable web service integration** flag in Figure 5.7 to **Yes**. By default, this is set to **No**.
6. From the **TT system** drop-down, select **Remedy Force** as the TT system with which the eG manager should integrate.
7. Then, to enable the eG manager to connect to the RemedyForce installation in your environment, specify the following:
   - **URL:** The URL using which the eG manager should connect to RemedyForce.
   - **Partner URL**: The partner URL to which the eG manager should first connect to connect to RemedyForce.
   - **User** and **Password**: The credentials of a valid user who has rights to access RemedyForce through the **Partner URL**.
8. The implementation of eG integration with RemedyForce exists in a Java class file named **SalesForceIntegrator.java**. Additionally, the API files and library files that support this class file are jarred in **EG-webservice.jar** file. In eG, this jar file is referred to as the **TT integration archive file**. By default, the implementation class file and the jar file are bundled with the eG manager and are available in the **<EG_INSTALL_DIR>\manager\lib** directory on the eG manager host. However, if later, changes are made to these files, you will have to re-upload them to the eG manager. In such a case, set the **Do you want to upload the supporting archives?** flag to **Yes**, and click the **Upload** button alongside. This will invoke Figure 5.8.

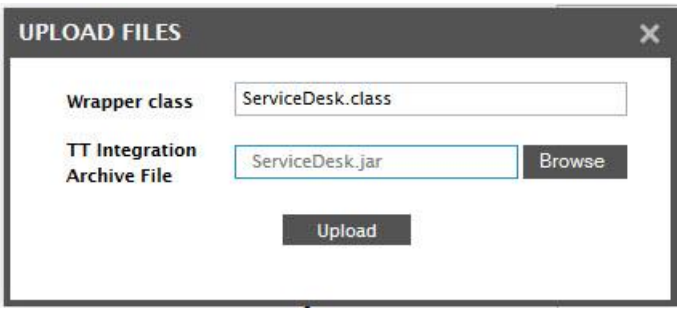

Figure 5.8: Uploading the TT integration jar file to the eG manager

Here, specify the full path to the **Wrapper class** file – i.e., the **SalesForceIntegrator.java** file - and the **TT Integration Archive File** – i.e., the **EG-webservice.jar** file - to be uploaded. Use the **Browse** button for the path specification. Finally, click the **Upload** button in Figure 5.8.

9. Finally, click the **Update** button in Figure 5.7.

5.5 Integration with PagerDuty

PagerDuty provides alerting, on-call scheduling, escalation policies and incident tracking to increase uptime of your apps, servers, websites and databases.

To ensure that eG integrates with PagerDuty, do the following:

1. Login to the eG administrative interface as *admin* with password *admin*.
2. Invoke the **Admin** tile menu and select the **Manager** option from the **Settings** tile.
3. In the **MANAGER SETTINGS** tree-structure in the left panel, expand the **Trouble Ticket Integration** node.
4. Next, select the **Web services** sub-node within. The right panel will change as shown by Figure 5.9 below.

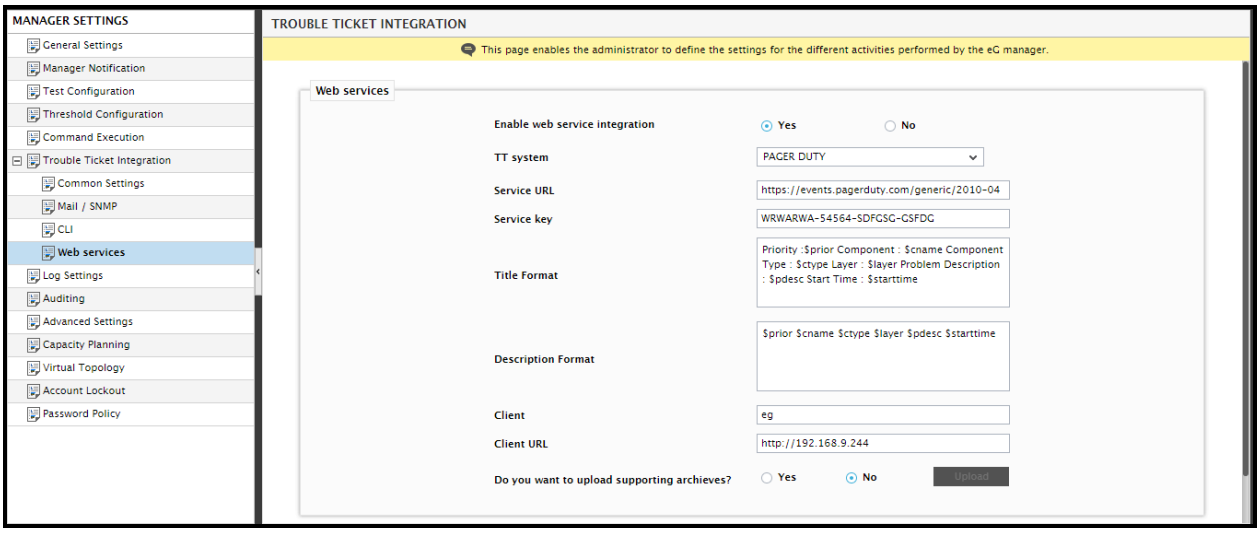

Figure 5.9: Integrating the eG manager with PagerDuty

5. To enable integration via the web services interface of PagerDuty, set the **Enable web service integration** flag in Figure 5.9 to **Yes**. By default, this is set to **No**.

6. From the **TT system** drop-down, select **PagerDuty** as the TT system with which the eG manager should integrate.

7. Then, to enable the eG manager to connect to the RemedyForce installation in your environment, specify the following:

   - **Service URL**: The web services URL using which the eG manager should connect to PagerDuty.
   - **Service key**: The unique key that is required for accessing the web services API of PagerDuty.
   - **Ticket title format: S**pecify the format in which the title of the trouble ticket is to be displayed in the PagerDuty console. The default format is as follows:

```
Priority :$prior Component : $cname Component Type : $ctype Layer : $layer
Problem Description : $pdesc Start Time : $starttime
```

   The text preceding the ':' (colon) in the format above indicates what information follows. The 'dollared' ($) text that follows the ':' (colon) is a variable, the value of which varies at run time, depending upon the information contained in the eG alarms. For example, in the default format above, *Priority* is a label that indicates that the information that follows the ':' is the priority of the alarm. The variable *$prior* that succeeds the ':' represents the alarm priority, and changes according to the priority of the actual alarm that is sent by the eG manager to PagerDuty. **While you can change the labels, you are advised against changing any of the variable names.**

   The other variables that are part of the default format are discussed in the table below:

$cname	Will display the name of the problem component
$ctype	Will display the component type to which the problem component belongs
$layer	Will display the layer affected by the problem
$pdesc	Will display a brief problem description
$starttime	Will display the start time of the problem

   - **Description format: S**pecify the format in which the problem description should be displayed in the trouble tickets generated by PagerDuty. The default problem description contains the following variables:

$prior	Refers to the problem priority
$cname	Refers to the name of the problem component
$ctype	Refers to the component type to which the problem component belongs
$layer	Refers to the layer affected by the problem
$pdesc	Refers to a brief problem description
$starttime	Refers to the start time of the problem

   You can override this default format by removing one/more variables or by changing the positions of a few variables. For instance, you may not want the problem component type to be part of the problem description. In this case therefore, your **Description format** will be.

```
$prior $cname $layer $pdesc $starttime
```

- **Client:** You may want the trouble ticket generated by PagerDuty to indicate which client supplied the problem information. If so, then provide the client name here. In the case of this integration, you may want to provide a unique identifier for the eG manager against **Client**.
- **Client URL**: You may want the trouble ticket generated by PagerDuty to indicate the URL of the client that supplied the problem information. If so, then provide the client URL here. In the case of this integration, you may want to provide the URL of the eG manager against **Client URL**.

8. The implementation of eG integration with PagerDuty exists in a Java class file named **PagerDutyntegrator.java**. By default, the implementation class file is bundled with the eG manager and is available in the **<EG_INSTALL_DIR>\manager\lib** directory on the eG manager host. However, if later, changes are made to this class file, you will have to re-upload it to the eG manager. In such a case, set the **Do you want to upload the supporting archives?** flag to **Yes**, and click the **Upload** button alongside. This will invoke an **UPLOAD FILES** window, as depicted by Figure 5.8. In this window, specify the full path to the **Wrapper class** file – i.e., the **PagerDutyIntegrator.java** file - to be uploaded. Use the **Browse** button for the path specification. Finally, click the **Upload** button in Figure 5.8.
9. Finally, click the **Update** button in Figure 5.9.

5.6 Integrating with any Third-party TT System that Supports a Web Services Interface

eG Enterprise provides a **TT Integration API**, which can be leveraged to enable the eG manager to integrate with any third-party TT system that supports a web services interface.

The broad steps to this end are as following:

1. Add the new TT system to the eG manager.
2. Indicate what parameters need to be configured for implementing the integration.
3. Using the **TT Integration API**, create a new **TT Integrator** class, where the implementation logic of the integration with the new TT system will be defined.
4. Jar the supporting files (if any) of the class.
5. Upload the TT integrator class and the supporting jar file (if any).
6. Restart the eG manager.

The sections that follow will discuss the first 5 steps of the integration process.

5.6.1 Adding the New TT System to the eG Manager

Once the TT integrator class and its supporting files are defined, proceed to add the new TT system to the eG manager.

1. For this, edit the **eg_services.ini** file in the **<EG_INSTALL_DIR>\manager\config** directory.

2. In the **[TT_INTEGRATION]** section of the file, you will find a **TT_INTEGRATION_SYSTEMS** parameter.
3. By default, this parameter will be configured with the comma-separated list of TT systems the eG manager integrates with out-of-the-box. Append the new TT system to this list. For instance, say that the new TT system is named *HelpDesk*, then the **TT_INTEGRATION_SYSTEMS** specification will be as follows:

   TT_INTEGRATION_SYSTEMS= SERVICE NOW,MANAGE ENGINE,AUTO TASK,REMEDY FORCE,HELPDESK
4. Finally, save the file.

Once this is done, *HELPDESK* will be automatically available for selection in the **TT system** drop-down of the **TT INTEGRATION – Web Services** page (see Figure 5.7) in the eG admin interface.

5.6.2 Defining the Parameters Required for Integrating with the New TT System

Next, specify what parameters need to be configured for integrating with the new TT system. This needs to be done so that the eG manager knows what parameters to display and make available for configuration in Figure 5.7, once the new TT system is chosen from the **TT system** drop-down of Figure 5.7. Typically, these parameters will vary from one TT system to another. Some of the most common parameters are as follows:

- The web services URL
- The permissions for accessing the TT system
- The format in which eG alarms are to be sent to the TT system
- SLAs (if any)

To define such parameters, do the following:

1. Edit the **eg_services.ini** file in the **<EG_INSTALL_DIR>\manager\config** directory.
2. In the **[TT_INTEGRATION_SYSTEM_FIELDS]** section of the file, insert an entry of the following format:

   <TT_Integration_System>=<Parameter1>,<Parameter2>,<Parameter3>,….

   For instance, to integrate with a TT system named *HelpDesk*, the URL, access permissions, and output format may have to be configured. To indicate that these are the parameters to be configured, your specification should be as follows:

   HelpDesk=url,user,password,outputFormat

- Make sure that the parameters configured do not embed white spaces.
- If 'password' is one of the parameters, then make sure you name that parameter as *password* only. If not, then eG will not be able to encrypt the value of that parameter.

3. Next, configure the default values (if any) that each of these parameters should take. For this, create a new section in the **eg_services.ini** file. The name of the section should be of the following format: *[TT_INTEGRATION_<Name_of_TT_system_in_capital_letters>]*. For instance, if your TT system is called *HelpDesk*, then create a section named *[TT_INTEGRATION_HELPDESK]*.

4. Then, insert entries of the following format for every parameter for which you want to configure a default value:

   <Parameter>=<Default_value>

   In the case of the *HelpDesk* example above, entries in the *[TT_INTEGRATION_HELPDESK]* section can be as follows:

   [TT_INTEGRATION_HELPDESK]

   url=www.helpdesk.com

   outputformat=<Operation>|n<Details>|n<requester>eG_manager</requester>|n<subject>$cname/$ctype/$pdesc/$prior</subject>|n<description>$pdesc</description>

   

   - Make sure that the parameter names specified in this section match with the corresponding parameter names in the **[TT_INTEGRATION_SYSTEM_FIELDS]** section. For instance, you cannot refer to a parameter that has been named as *url* in the **[TT_INTEGRATION_SYSTEM_FIELDS]** section as *uri* in this section.
   - If one/more parameters in the **[TT_INTEGRATION_SYSTEM_FIELDS]** section are excluded from the *[TT_INTEGRATION_<Name_of_TT_system_in_capital_letters>]* section, it implies that the excluded parameters do not take any default value. In this case, no value will be displayed by default for these parameters in the **TT INTEGRATION – Web Services** page (see Figure 5.7).
   - Optionally, you can set *$unconfigured* as the default value of a parameter in the *[TT_INTEGRATION_<Name_of_TT_system_in_capital_letters>]* section. In this case, the **TT INTEGRATION – Web Services** page too will display the value *$unconfigured* by default for that parameter. This implies that the parameter does not take any default value, but a value should be provided for that parameter in the **TT INTEGRATION – Web Services** page.

5. Now that the default values are also set, proceed to indicate what type of form control should be mapped to each parameter. A control of the specified type will then appear against that parameter in the **TT INTEGRATION – Web Services** page (see Figure 5.7). A parameter can support only one of the following three form controls: *TextField, TextArea, Combo*. To indicate which control maps to which parameter, insert an entry of the following format for every parameter defined for the new TT system, in the [**TT_INTEGRATION_FIELD_MAPPING]** section of the **eg_services.ini** file.

   [TT_INTEGRATION_FIELD_MAPPING]

   <Parameter>=<Control_Type>

   By default, entries will pre-exist in the [**TT_INTEGRATION_FIELD_MAPPING]** section for common parameters such as *url, user, password, outputformat,* etc. In such a case, insert entries in this section for only those parameters that are unique to the new TT system. For instance, assume that a unique parameter named *zone* is configured for a TT system. To make sure that this parameter is to be displayed as a combo box in the eG admin interface, insert the following entry in the [**TT_INTEGRATION_FIELD_MAPPING]** section.

   zone=Combo

- **You are advised not to make changes to the specifications that pre-exist in the [TT_INTEGRATION_FIELD_MAPPING] section**. This is because, it is likely that parameters for which mappings are defined here are shared by multiple TT systems; so, changes effected here may impact what controls are displayed for these parameters across all related TT systems.
- Make sure that there is an entry in this section for every parameter that you have defined for the new TT system in the **[TT_INTEGRATION_SYSTEM_FIELDS]** section. If not, such a parameter will not even appear in the **TT INTEGRATION – Web Services** page of Figure 5.7.

6. Now, indicate what meaningful label (or display name) is to be used to identify each parameter configured for the new TT system in the **[TT_INTEGRATION_SYSTEM_FIELDS]** section. For this, insert an entry of the following format for each parameter of a TT system, in the **[TT_INTEGRATION_TEXT_MAPPING]** section:

   <Parameter>=<Label>

   A sample specification has been provided below:

   [TT_INTEGRATION_TEXT_MAPPING]

   url=URL

   user=User name

   password=Password

   outputformat=Output Format

   The labels specified in this section will appear as field labels in the **TT INTEGRATION – Web Services** page of Figure 5.7 for the chosen **TT system.**

- By default, entries will pre-exist in the [**TT_INTEGRATION_TEXT_MAPPING]** section for common parameters such as *url, user, password, outputformat,* etc. Changing the label for any of these parameters will change that label for all TT systems with which that parameter is mapped.
- If one/more parameters in the **[TT_INTEGRATION_SYSTEM_FIELDS]** section are excluded from the **[TT_INTEGRATION_TEXT_MAPPING]** section, then for such parameters, the parameter names themselves will be displayed as field labels in the **TT INTEGRATION – Web Services** page (see Figure 5.7).

7. Finally, save the **eg_services.ini** file.

5.6.3 Creating a TT Integrator Class for the New TT system

Since the framework for the integration has now been defined, its time to create the TT integrator class.

5.6.3.1 System Requirements

The **TT Integration API** is included as part of the eG manager package. In order to use the API for building the **TT Integrator** class, you must first set the CLASSPATH environment setting for the user who is creating the class so as to include the Java archive file, **<EG_MANAGER_HOME_DIR>\manager\lib\egTroubleTicketing.jar**.

5.6.3.2 Component Classes

The TT Integration API Consists of a single Abstract class called **TTIntegrationWrapper** whose functionality has to be implemented in order to develop the TT Integrator class.

5.6.3.3 Summary of Methods

The following methods of the **TT Integration** API must be used for developing new tests.

Method Signature	Description
Public void updateInitParams(Hashtable initParams)	This method is called to update the TT integrator class with the details of which TT system to connect to, how to connect to it (i.e., which URL is to be hit), and what user permissions are required for the connection, whenever changes are made to one/more of these parameters by a user.
Public String createTicket(Object[] alertDetails) throws Exception	This method sends the details of an alarm raised by the eG manager to the TT system and requests for the creation of a trouble ticket. Based on the response received from the TT system- which can either be a trouble ticket ID or an error message – the method then maps the trouble ticket ID to the eG alarm or logs the error message (as the case may be). **Description of arguments:** `Object[] alertDetails`: This is an array list that will contain the details of the alarm to be sent to the TT system.
Public boolean updateTicket(String ticketID,Object[] alertDetails) throws Exception	If an eG alarm changes, then this method updates the corresponding trouble ticket with the changes. An alarm change can be one/all of the following: • A change in alarm priority; • A change in alarm description: • A change in the list of impacted services Based on the response received from the TT system - which can either be an acknowledgement that the trouble ticket has been successfully updated or an error message – the method then maps the trouble ticket ID to the eG alarm or logs the error message. **Description of arguments:** `ticketID`: This argument represents the ID of the trouble ticket that has changed. `Object[] alertDetails`: This is an array list that will contain the details of the alarm changes with which the `ticketID` is to be updated.

5.6.3.4 Writing the TT Integrator Class using the TT Integration API

The **TT Integrator** class must extend **TTIntegrationWrapper** class and override the abstract methods to implement the integration.

Given below is a sample TT Integrator class that sends eG alerts to the TT System and creates / updates / closes tickets.

```
package eGTTIntegration;

public class TestIntegrator extends TTIntegrationWrapper {
	/* Declare the necessary Global Variables */
	String url= "";
	String username = "";
	String password = "";
	String hostPort= "";

	private DefaultHttpClient client = null;

	// @param InitParam holds initial config params such as url,username,password
needed of the TT System.
	public TestIntegrator (Hashtable initParams) {
		updateInitParams(initParams);
	}
	/**
	* Used to Update Init params
	*
	* @param initParams
	*
	*/
	public void updateInitParams(initParams) {
		this.url=(String)initParams.put("url");
		this.username=(String)initParams.put("username");
		this.password=(String)initParams.put("password");
		this.hostPort=(String)initParams.put("hostPort");
	}
	/* used to connect web service */

	private void connect()	{
// write the code to connect web service
	}

	/*
	 * This method used to create the ticket

	 * @param problemDetails probDetails
	 */
	public String createTicket(Object[] probDetails) throws Exception {
//		call connect method here
// get the required information from prob Details to create a ticket and send the same
thru webservice
// receive the Ticket Id and return the same
		return ticketId;
	}
	/*
	 * This method used to update the ticket

	 * @param ticket Id,problem Details.
```

```
        */
       public boolean updateTicket(String ticketId,Object[] probDetails) throws
IOException {
              //           call connect method here
// get the required information from prob Details to update a ticket and send the same
thru webservice
// receive the response and check the status.if update successful return true
otherwise return false;
              return isUpdated;
       }
}
```

5.6.4 Jarring the Supporting Files

Sometimes, your TT integrator class may borrow functionality from DLLs or other supporting files that may not be available on the eG manager already. For the integration to occur smoothly, you will have to jar these supporting files and upload this jar file along with the TT integrator class to the eG manager. For this purpose, first jar all supporting files of the TT integrator class.

5.6.5 Uploading the TT Integrator Class and Supporting Files to the eG Manager

For this purpose, do the following:

1. Login to the eG administrative interface as *admin* with password *admin*.
2. Invoke the **Admin** tile menu and select the **Manager** option from the **Settings** tile.
3. In the **MANAGER SETTINGS** tree-structure in the left panel of Figure 5.10, expand the **Trouble Ticket Integration** node.
4. Next, select the **Web services** sub-node within.

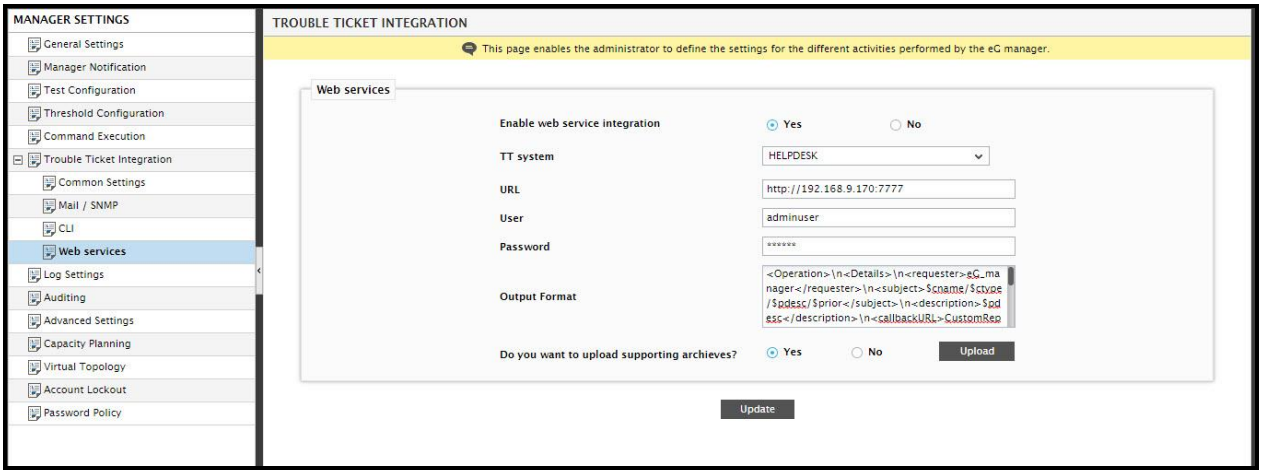

Figure 5.10: Integrating the eG manager with a Third-party TT system that supports a web services framework

5. To enable integration via the web services interface of the new TT system that you added (using the procedure detailed in Section 5.6.1 of this procedure, set the **Enable web service integration** flag in Figure 5.10 to **Yes** and select the new TT system from the **TT system** drop-down.

6. Then, provide values for all the parameters that you had configured for the new TT system in Section 5.6.2 above.

7. To upload the TT integrator class and supporting jar file, set the **Do you want to upload support archives?** Flag in Figure 5.10 to **Yes**. Then, click the **Upload** button alongside.

8. Figure 5.11 will then appear. Use the **Browse** button against **Wrapper class** in Figure 5.11 to locate the TT integration wrapper class that you had written for integrating with the new TT system (refer to Section 5.6.3). Likewise, use the **Browse** button against **TT Integration Archive File** in Figure 5.11 to locate the supporting files that you had jarred (refer to Section 5.6.4).

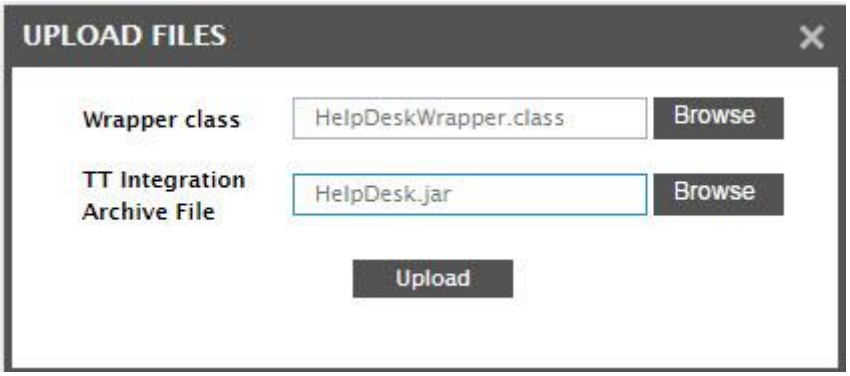

Figure 5.11: Uploading the TT integration wrapper class and supporting jar file of the new TT system

To upload both the archive file and the wrapper class, click the **Upload** button in Figure 5.11.

9. After successfully uploading the file, click the **Update** button in Figure 5.10.

Finally, restart the eG manager. With that, we have successfully integrated the eG manager with a new TT system.

6

Conclusion

The eG Enterprise Suite has been specially designed keeping in mind the unique requirements of IT infrastructure operators. For more information on the eG family of products, please visit our web site at www.eginnovations.com.

For more details regarding eG Enterprise suite of products and the details of the metrics collected by the eG agents, please refer to the following documents:

- *Administering the eG Enterprise Suite*
- *Monitoring eG Enterprise*
- *The eG Installation Guide*
- *The eG Measurements Manuals*

We recognize that the success of any product depends on its ability to address real customer needs, and are eager to hear from you regarding requests for enhancements to the products, suggestions for modifications to the product, and feedback regarding what works and what does not. Please provide all your inputs as well as any bug reports via email to sales@eginnovations.com.