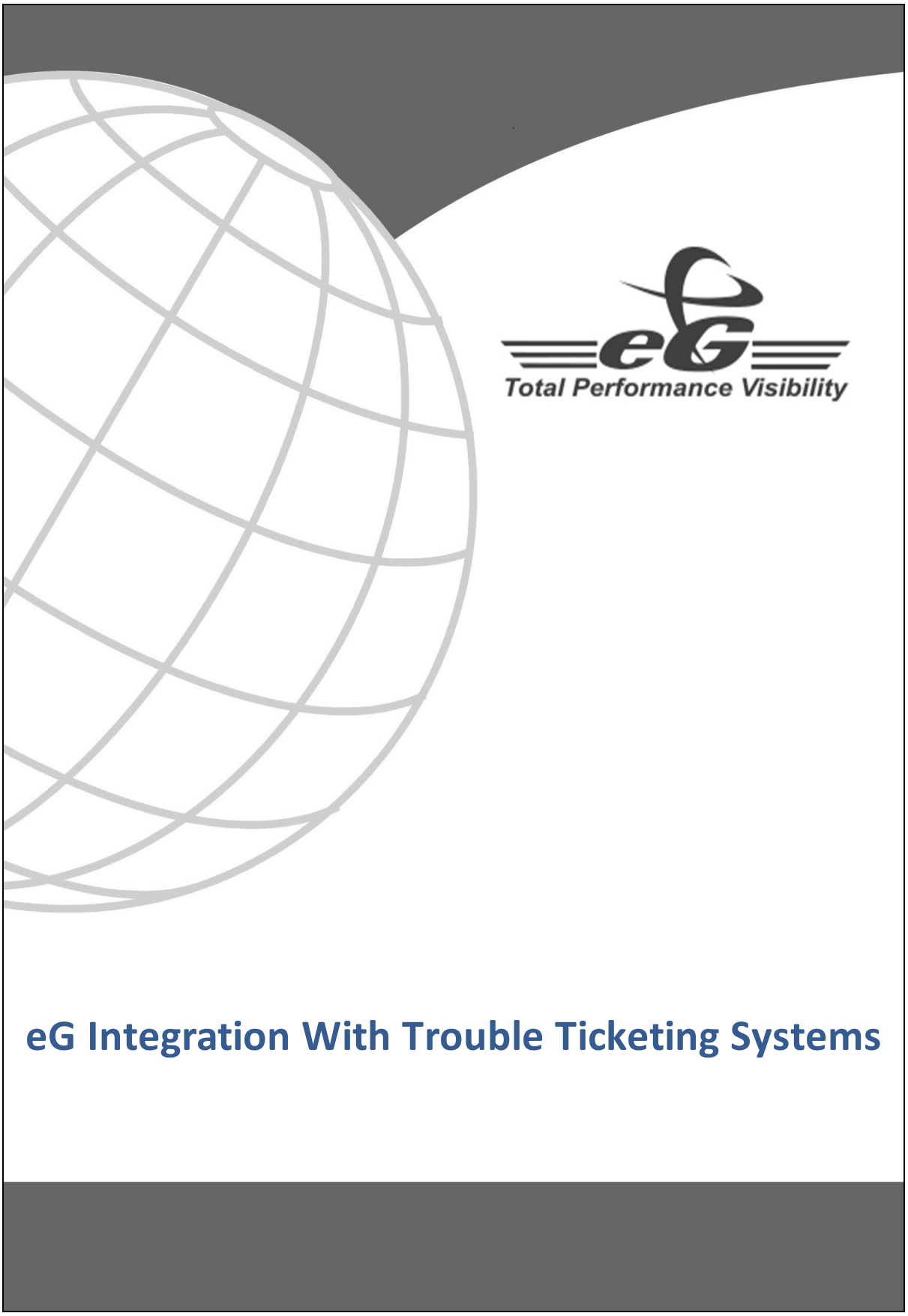

eG Integration With Trouble Ticketing Systems

Restricted Rights Legend

Trademarks

Copyright

Table of Contents

Table of Figures

Chapter 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. For the detailed discussion on this, refer to **Trouble Ticket Integration Using the TT Mail Interface** chapter.
- 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. To know how this is done, refer to **Trouble Ticket Integration Using SNMP Traps** chapter.
- 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. **Trouble Ticket Integration Using the eG TT CLI** chapter discusses this in great detail.
- 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. See **Trouble Ticket Integration Using a Web Services Framework**.

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.

Note:

- 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.

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.

1.1.4 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. For the detailed discussion on this, refer to **Trouble Ticket Integration Using the TT Mail Interface** chapter.
- 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. To know how this is done, refer to **Trouble Ticket Integration Using SNMP Traps** chapter.
- 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. **Trouble Ticket Integration Using the eG TT CLI**chapter discusses this in great detail.
- 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. See **Trouble Ticket Integration Using a Web Services Framework**.

1.1.5 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.

Note:

- 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.

Chapter 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.

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 eG Enterprise* 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, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

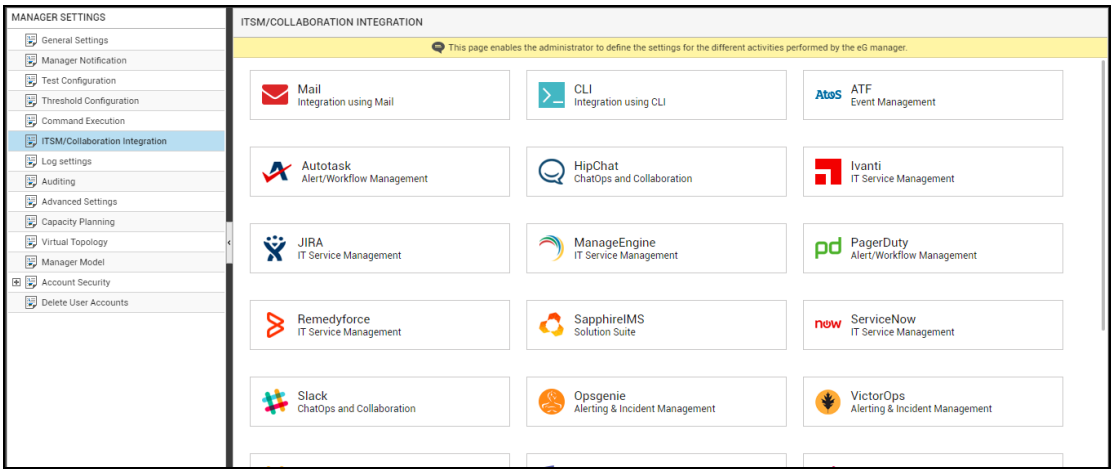

Figure 2.1: Viewing the ITSM/Collaboration tool options

4. Now, click on the **Mail** option in the right panel (see Figure 2.1). A **Mail** section will now appear in the right panel (see Figure 2.2).

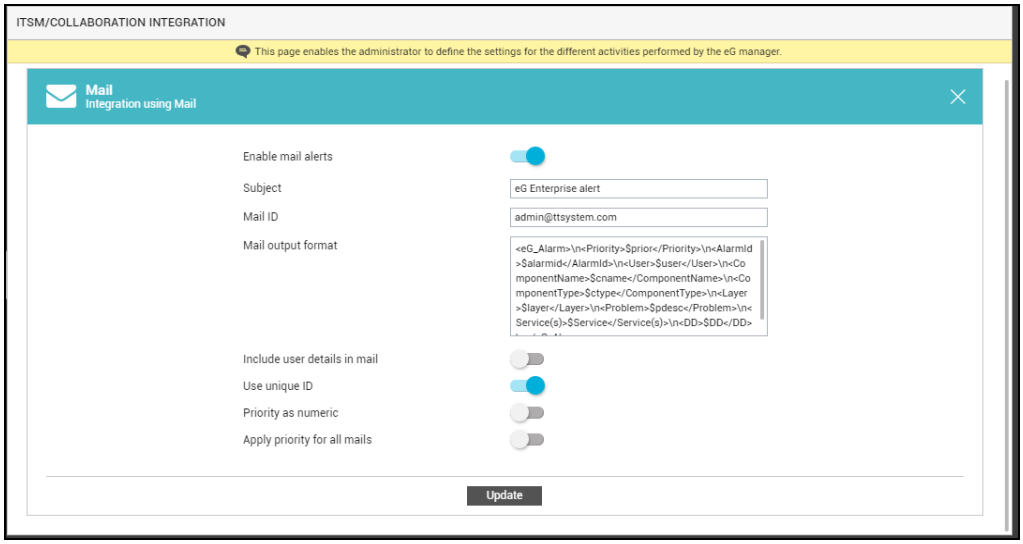

Figure 2.2: Configuring the TT mail settings

5. To enable TT integration using a mail interface, slide the **Enable mail alerts** slider in the Figure 2.2 to the left.

6. Next, 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:

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 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.

7. Next, against the **Mail ID** field, provide a comma-separated list of email IDs to which the email alerts are to be delivered.
8. 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>$ctype</ComponentType>\n<Layer>$
layer</Layer>\n<Problem>$pdesc</Problem>\n<Service(s)>$Service</Service
(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>
```

9. 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, slide the **Include user details in mail** slider to the right. By default, this flag is turned off .

10. 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, slide the **Use unique ID** slider to the right. By default, this flag is turned off.

11. 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 slide the **Priority as numeric** slider in 2.2 to the right. By default, this flag is turned off, indicating that the priority of an alarm is indicated using the priority name by default. If this flag is turned on instead, then the default priority name-number mappings defined in the **[TTMAIL]** section of the **eg_services.ini** file (in the <EG_INSTALL_DIR>\manager\config folder) will automatically apply.

    According to these mappings, if the **Priority as numeric** flag turned on, 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
    ```

12. Also, by toggling the **Apply priority for all TT mails** flag, you can indicate whether the **Alarm preference** setting (see Figure 2) 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** slider is moved to the right, 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 preference** in the **Settings** page (see Figure 2). On the other hand, if the **Apply priority for all TT mails** slider is moved to the left, 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.

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

Note:

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 **SystemDetails** 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
```

Chapter 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 Sending Trouble Tickets over SNMP Traps to know how to configure the eG manager so.

3.1 How to Enable TT Integration over SNMP Traps?

There are two broad steps for enabling the TT integration over SNMP traps:

- Configure a Third-party SNMP Manager
- Send Trouble Tickets over SNMP Traps

These steps have been elaborately explained in relevant sections.

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.

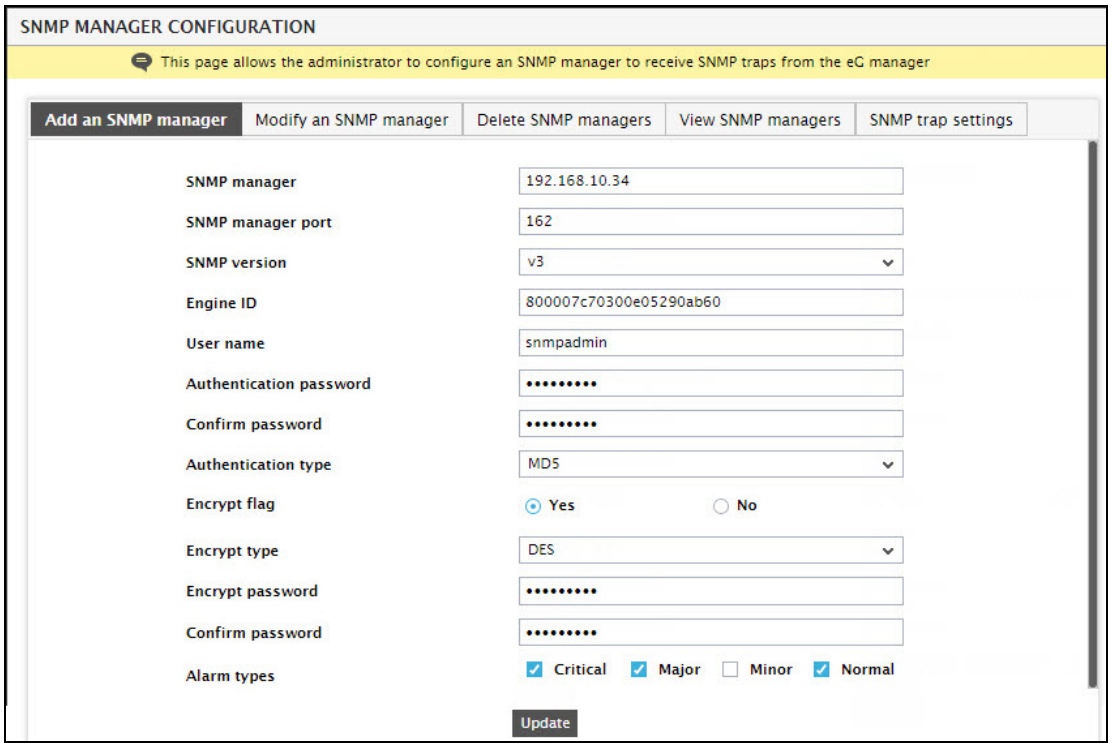

Figure 3.1: Adding an SNMP manager

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.

7. 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.
8. Finally, click the **Update** button to add the new SNMP manager.
9. 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.

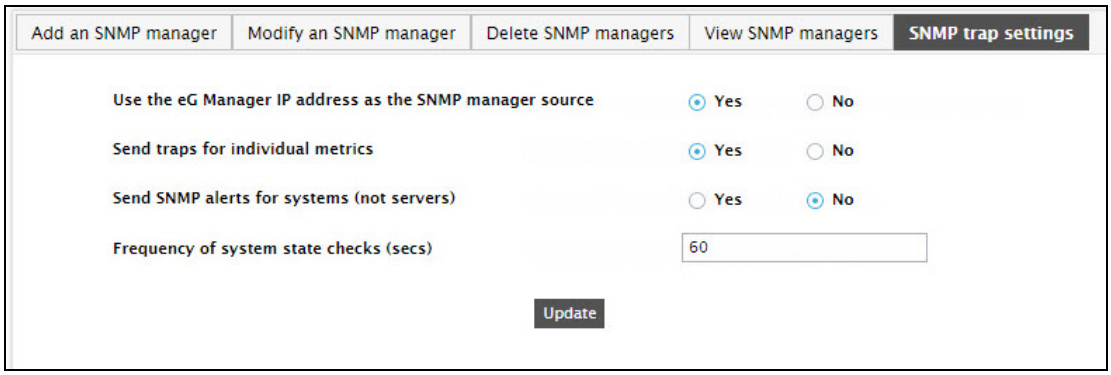

Figure 3.2: Configuring the SNMP trap settings

10. 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.
11. 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**.

12. 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 this flag to **No**.
13. 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).
14. 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.

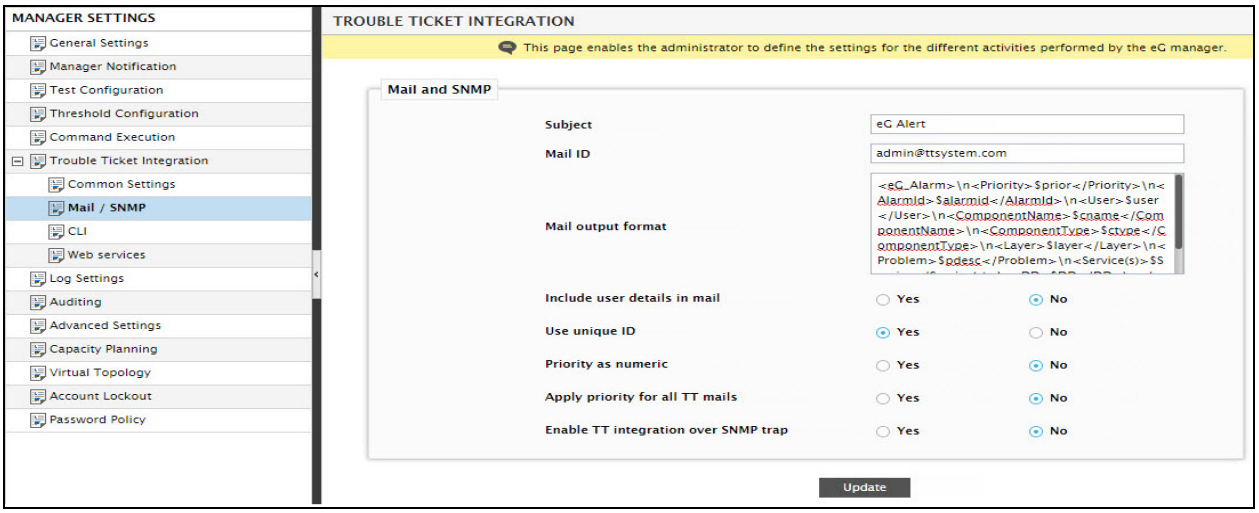

Figure 3.3: Enabling TT integration over SNMP traps

3. Set the **Enabling TT integration over SNMP traps** flag to **Yes**.
4. Finally, click the **Update** button.
5. 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 each 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

Note:

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. Also, such traps will not be affected by the **Alarm types** and **SNMP Trap Settings** configured using Figure 3.1 and Figure 3.2.

Note:

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, 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. 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 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.

Chapter 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.

To configure this capability, do the following:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 4.1 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 4.1, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

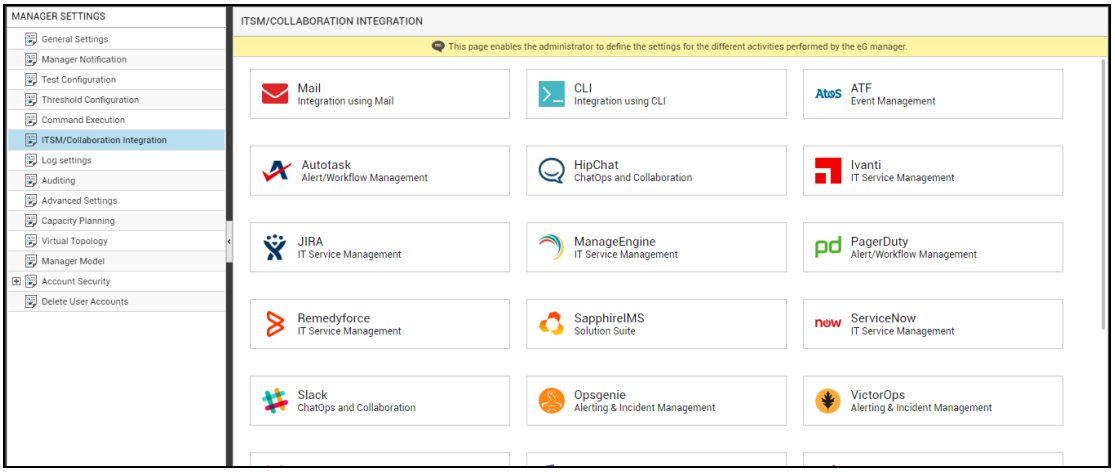

Figure 4.1: Viewing the ITSM/Collaboration tool options

4. Now, click on the **CLI**option in the right panel (see Figure 4.1). A **CLI** section will now appear in the right panel (see Figure 4.2).

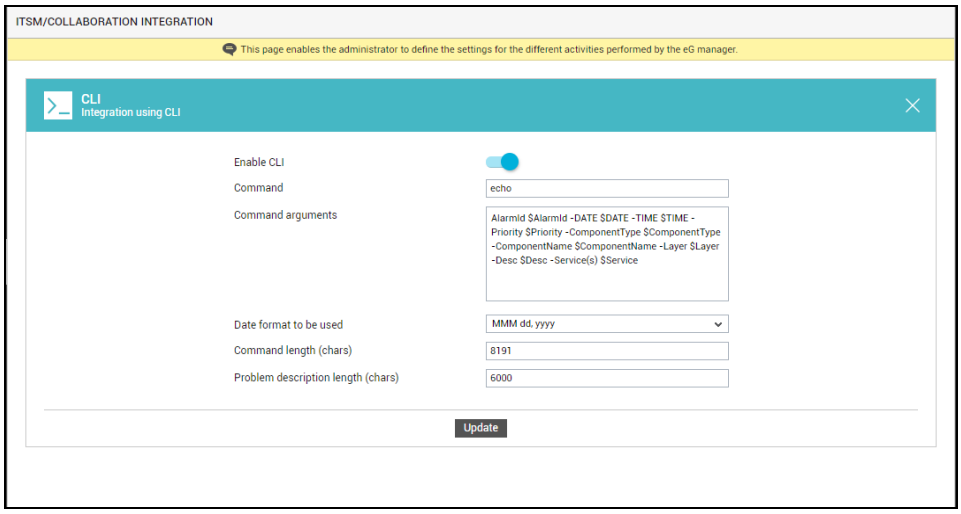

Figure 4.2: Configuring the TT mail settings

5. Slide the **Enable CLI** slider (see Figure 2) to the right to enable this capability.
6. 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.
7. 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, 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*.

Note:

- 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 DDcolumn… ~#~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.

Note:

- 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.

```
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:

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

Qualifier	Variable
AlarmId	192.168.10.133_1256878963888
DATE	30/10/2009

Qualifier	Variable
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-

Qualifier	Variable
	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

Qualifier	Variable
	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

Note:

- 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.

8. 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.
9. 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.

10. 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.
11. 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).

Chapter 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
- HipChat
- Slack
- JIRA

Section **5.1** to Section **5.8** that follow will discuss how eG integrates with each of the TT systems mentioned above.

The topics will discuss how eG integrates with each of the TT systems mentioned above.

- Section **5.1**
- Section **5.2**
- Section **5.3**
- Section **5.4**
- Section **5.5**
- Section **5.6**
- Section **5.7**
- Section **5.8**

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.

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.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.1 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.1, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

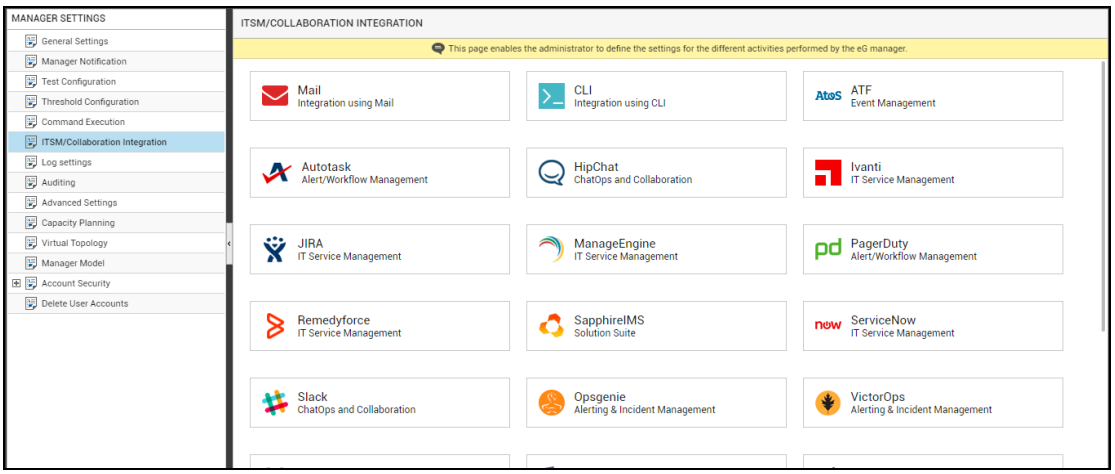

Figure 5.1: Viewing the ITSM/Collaboration tool options

4. Now, click on the **ManageEngine**option in the right panel (see Figure 5.1). A **ManageEngine** section will now appear in the right panel (see Figure 5.2).

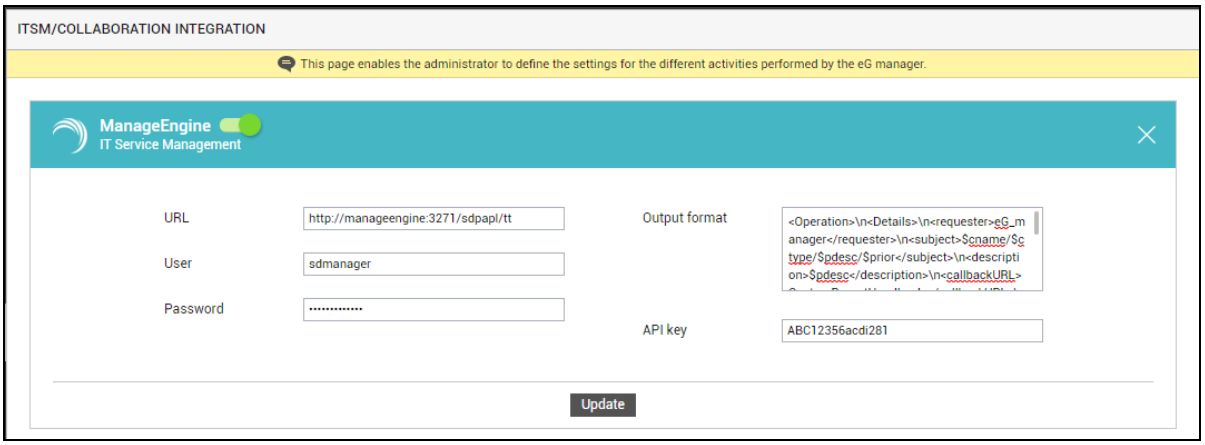

Figure 5.2: Configuring integration with ManageEngine ServiceDesk

5. To enable integration with ManageEngine ServiceDesk, first slide the **ManageEngine** slider in Figure 5.2 to the right.
6. Then, against **URL**, specify the Web Services Description Language (WSDL) URL via which the eG manager should connect to ServiceDesk's web services interface.
7. If the connection needs to be authenticated, then provide a valid user name and password against **User** and **Password** text boxes, respectively.
8. 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 Output format in 5.1, is as follows:

```
<Operation>\n<Details>\n<requester>eG_
manager</requester>\n<subject>$cname/$ctype/$pdesc/$prior</subject>\n<description>$pde
sc</description>\n<callbackURL>CustomReportHandler.do</callbackURL>\n<requesttemplat
e>-
</requesttemplate>\n<priority>$prior</priority>\n<layer>$layer</layer>\n<group>$user</
group>\n<technician>eG_
manager</technician>\n<level>1</level>\n<status>$status</status>\n<service>$Service</s
ervice>\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 'keys' representing the subject of the trouble ticket. In the case of the standard output format above, the following keys 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. • $prior – at run time, this variable will change to report the problem priority You can remove the keys at will, and can also alter the position of the keys in the output.
<description></description>	Should contain the key that represents the problem description – i.e., *$pdesc*. At run time, this key 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 key that represents the problem priority – i.e., *$prior*. At run time, this key will change to report the precise problem priority.
<layer</layer>	Should contain the key that represents the problem layer – i.e., *$layer*. At run time, this key will change to report the problematic layer.
<group></group>	Should contain the key that represents the user/support group to which the technician responsible for resolving this alarm belongs. At run time, this key 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 key **$status** that represents the status of the ticket. At run time, this key will change to report the exact status.
<service></service>	Should contain the key **$service** that represents the name of the business service impacted by the problem. At run time, this key will change to to report the correct business service name.

9. 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 5.2 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**.
10. 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.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.3 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.3, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

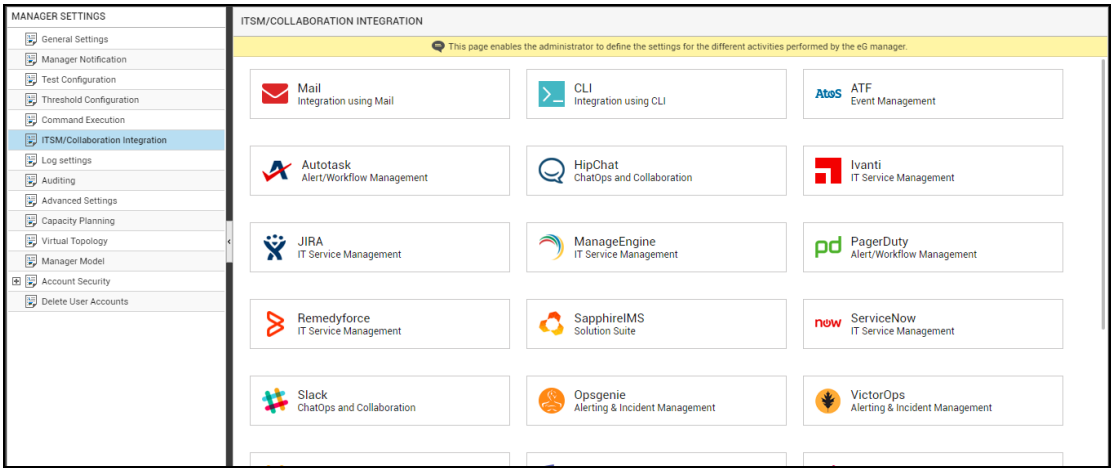

Figure 5.3: Viewing the ITSM/Collaboration tool options

4. Now, click on the **ServiceNow**option in the right panel (see Figure 5.3). A **ServiceNow** section will now appear in the right panel (see Figure 5.4).

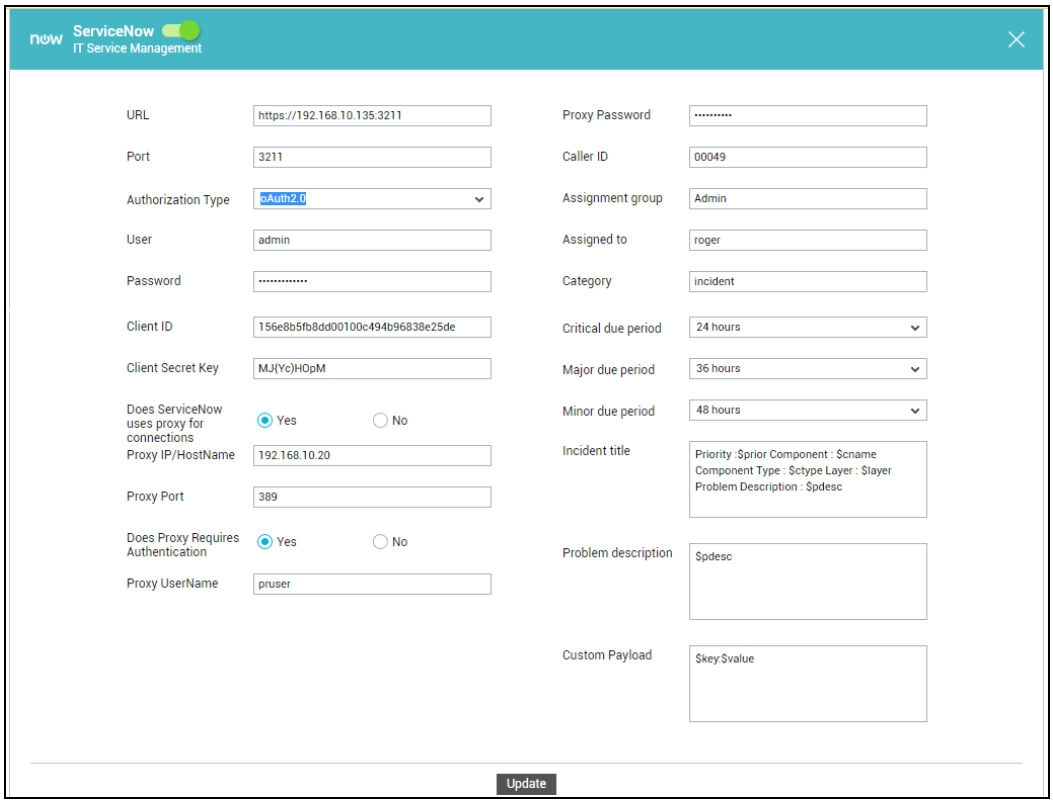

Figure 5.4: Configuring integration with ServiceNow

5. To enable integration with ServiceNow, first slide the **ServiceNow** slider in Figure 5.4 to the right.
6. Then, specify the following in Figure 5.4:

   - **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.
   - **Authorization Type**: The eG manager sends alarm information to ServiceNow as a web service request to the configured **URL**. Upon receipt of the request, ServiceNow will attempt to validate the source of the request using one of the following authentication methods:
     - Basic authentication
     - O Auth 2.0 authentication

     If ServiceNow enforces Basic Authentication, then select the **Basic** option from the **Authorization Type** drop-down. Where Basic Authentication is enforced, ServiceNow

requires that web service requests be accompanied by the username and password of a user who has access to the ServiceNow instance. Accordingly, if **Basic** is set as the **Authorization Type**, you need to provide the credentials of a user with the right to access ServiceNow in the **User** and **Password** text boxes.

On the other hand, if ServiceNow enforces the O Auth 2.0 authentication method, then select the **OAuth 2.0** option from the **Authorization Type** drop-down. O Auth 2.0 lets users access instance resources through external clients by obtaining a token rather than by entering credentials with each resource request. This means that where O Auth 2.0 is enforced, the eG manager needs to obtain an access token, so it can create/modify trouble tickets in ServiceNow. For this, the eG manager should first connect to the ServiceNow instance as a user who is authorized to request for an access token, and then submit web service requests as a valid 'Client'. This is why, if **OAuth 2.0** is set as the **Authorization Type**, you will have to specify the following:

- **User** and **Password**: The credentials of a user who is authorized to request for an access token
- **Client ID** and **Client Secret Key**: The **Client ID** is an auto-generated unique ID of the client application - i.e., in our case, the eG manager application - requesting the access token. The **Client Secret Key** is a shared secret string that the ServiceNow instance and the client applications - i.e., the eG manager - use to authorize communications with one another.

- **Does ServiceNow use proxy for connections**: If the eG manager needs to communicate with the ServiceNow instance via a Proxy server, then set this flag to **Yes**.
- **Proxy IP/Hostname** and **Proxy Port**: If the **Does ServiceNow use proxy flag** is set to **Yes**, then specify the IP/host name of the Proxy server and the port number at which the Proxy server listens in the respective text boxes.
- **Does Proxy require authentication**: This flag is applicable only if the **Does ServiceNow use proxy** flag is set to **Yes**. If so, then use this flag to indicate whether/not the Proxy server requires authentication. Set this flag to **Yes**, only if the Proxy server requires authentication.
- **Proxy UserName** and **Proxy Password**: If the **Does Proxy require authentication** flag is set to **Yes**, then provide the credentials of a valid Proxy user here.
- **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.
- **Incident title**: Next, using the **Incident title** text box, 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 key, 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 key $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 keys**.

The other keys 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
```

- **Problem description**: The eG manager sends alarm information to ServiceNow in the form of a JSON file. Every piece of information contained within an eG alert - eg., priority, component name, component type etc. - is represented in the JSON file as a $key:$value pair, where 'key' denotes the alert field, and 'value' denotes the actual value of that field at run time. In the **Problem description** text box, specify the exact key in the JSON file that is used to capture the problem description. By default, *$pdesc* is the key used for reporting the problem description to ServiceNow. **Note that the keys in the JSON file will match the keys supported by the ServiceNow REST API.**
- **Custom Payload**: Use custom payload to customize the alert information you send to ServiceNow, so that it includes additional static information.

  Typically, the details of an eG alert are sent as a JSON file to the configured URL. Every piece of information contained within an eG alert - eg., priority, component name, component type etc. - is represented in the JSON file as a $key:$value pair, where 'key' denotes the alert field, and 'value' denotes the actual value of that field at run time. The 'key' is configured based on the keys supported by the ServiceNow REST API. For instance, if the REST API represents alarm priorities using the key 'prior', then the same key will be used in the JSON file for denoting alarm priorities. Accordingly, the entry for alarm priority in the JSON file will be $prior:$value. The $value will be Critical, Major, Minor, or Normal, depending upon the actual priority of the alarm being sent.

  If you want eG incidents routed to ServiceNow to include additional information, then you can define a **Custom Payload** for that information as a *$key:$value* pair. For example, say, you want incidents to indicate the FQDN of the eG manager that generated the incidents. Say that the FQDN of your eG manager is *egmanager.innovations.com*. To include this information in ServiceNow incidents, do the following:

  - First, check whether the ServiceNow REST API supports a 'key' that can be used for capturing the 'source' of alerts/incidents. If no such key exists, then you cannot proceed with the Custom Payload configuration. On the other hand, if such a key is available, then proceed to replace the $key in your Custom Payload specification, with that key value. For the purpose of our example, let us assume that the REST API supports the key named 'source'. In this case therefore, substitute '$key' with 'source'.
  - Then, proceed to explicitly specify the FQDN of your eG manager in the place of $value. This is because, you can use the Custom Payload configuration to add only 'static' information - i.e., information that you explicitly configure, and hence will never change. In the case of our example therefore, the $value will be egmanager.innovations.com.

- The complete **Custom Payload** specification will now be: *source:egmanager.innovations.com*

7. Finally, click the **Update** button in 5.2.

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.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.5 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.5, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

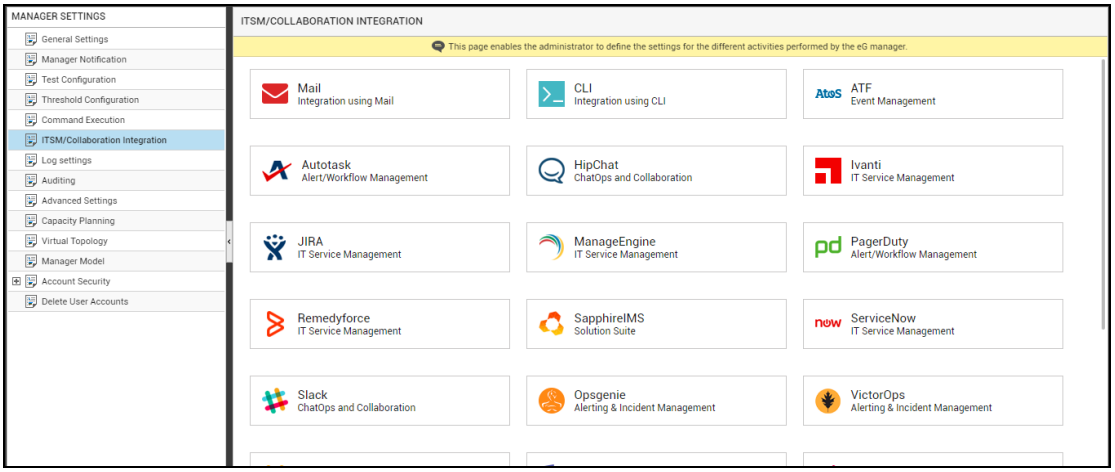

Figure 5.5: Viewing the ITSM/Collaboration tool options

4. Now, click on the **Autotask**option in the right panel (see Figure 5.5). An **Autotask** section will now appear in the right panel (see Figure 5.6).

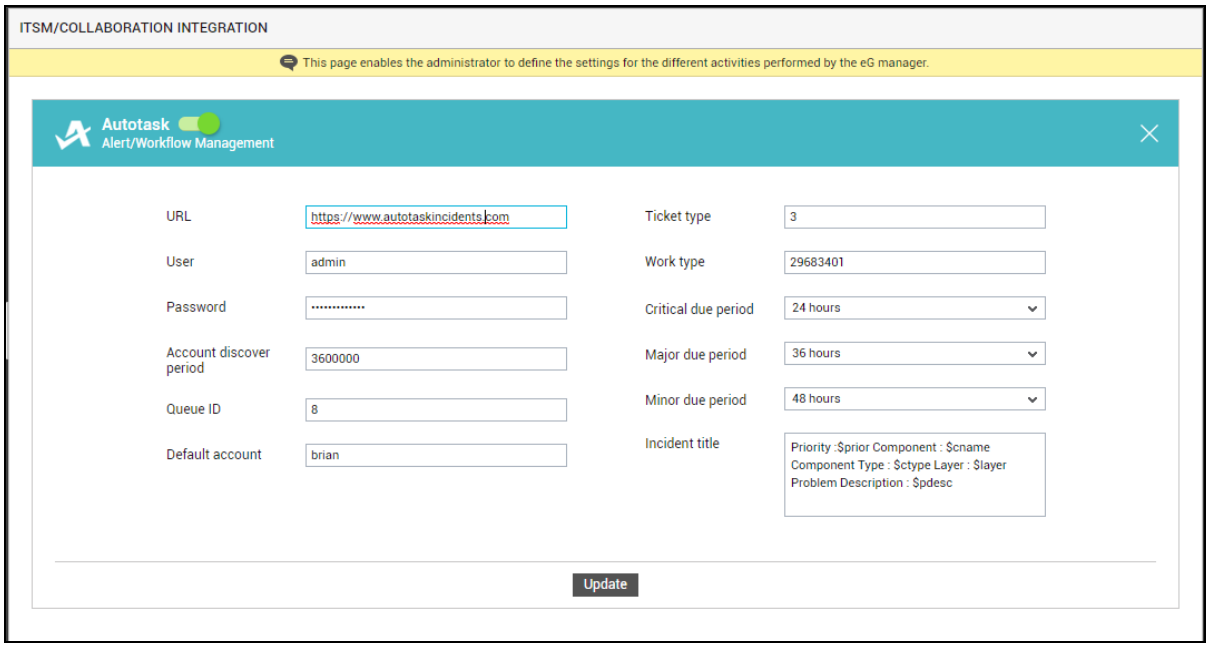

Figure 5.6: Configuring integration with Autotask

5. To enable integration with Autotask, first slide the **Autotask** slider in Figure 5.6 to the right.
6. Then, specify the **URL** using which the eG manager should connect to Autotask.
7. Next, against **User** and **Password**, provide the credentials of a valid user who has rights to access Autotask.
8. Then, in the **Account discover period** text box, 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.

9. In **QueueID**, 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.
10. Against **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.
11. In the **Ticket type** text box, 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.
12. In the **Work type** text box, specify the work type to be assigned to the trouble tickets generated for the eG alarms. By default, this is 29683401.
13. Next, specify the **Critical due period**. This is the time (in millisecs) by which tickets of a Critical priority will be resolved.
14. Then, indicate the **Major due period**, which is the time (in millisecs) by which tickets of a Major priority will be resolved.
15. Likewise, enter the **Minor due period**. This is the time (in millisecs) by which tickets of a Minor

priority will be resolved.

16. Next, against **Incident title**, 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
```

17. Finally, click the **Update** button in Figure 5.6.

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.
2. Select the **Manager** option from the **Settings** tile.

3. Figure 5.7 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.7, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

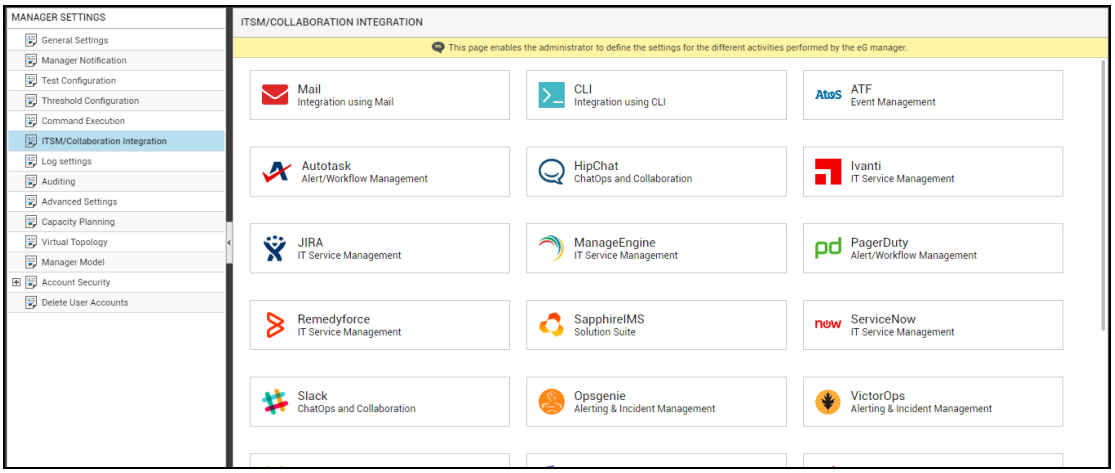

Figure 5.7: Viewing the ITSM/Collaboration tool options

4. Now, click on the **Remedyforce** option in the right panel (see Figure 5.7). A **Remedyforce** section will now appear in the right panel (see Figure 5.8).

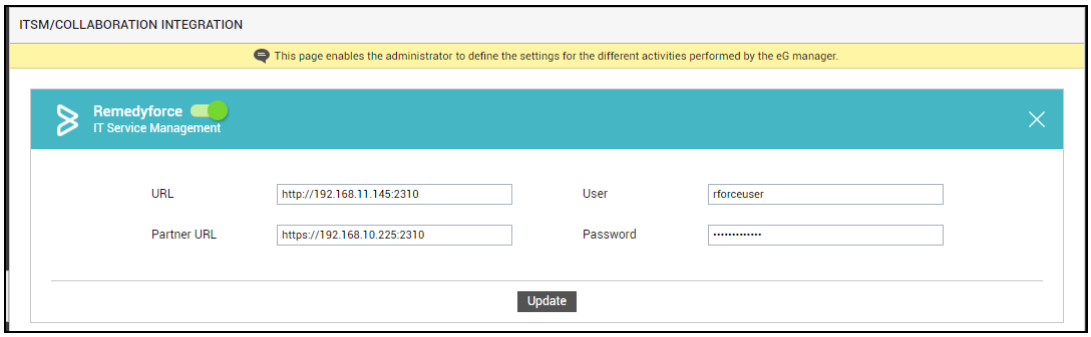

Figure 5.8: Configuring integration with Remedyforce

5. To enable integration with Remedyforce, first slide the **Remedyforce** slider in Figure 5.8 to the right.
6. Then, specify the following in Figure 5.8:

   - **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.

7. Finally, click the **Update** button in Figure 5.8.

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.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.9 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.9, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

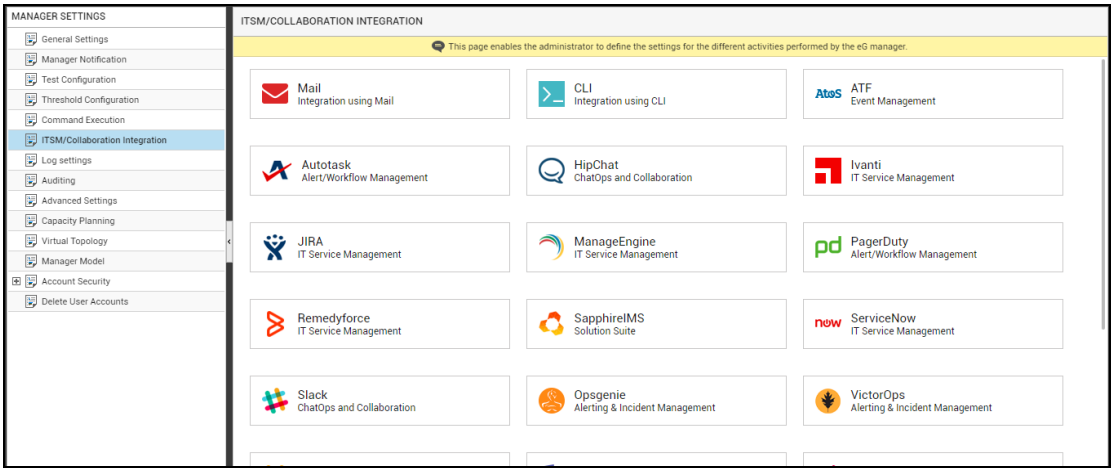

Figure 5.9: Viewing the ITSM/Collaboration tool options

4. Now, click on the **PagerDuty**option in the right panel (see Figure 5.9). A **PagerDuty** section will now appear in the right panel (see Figure 5.10).

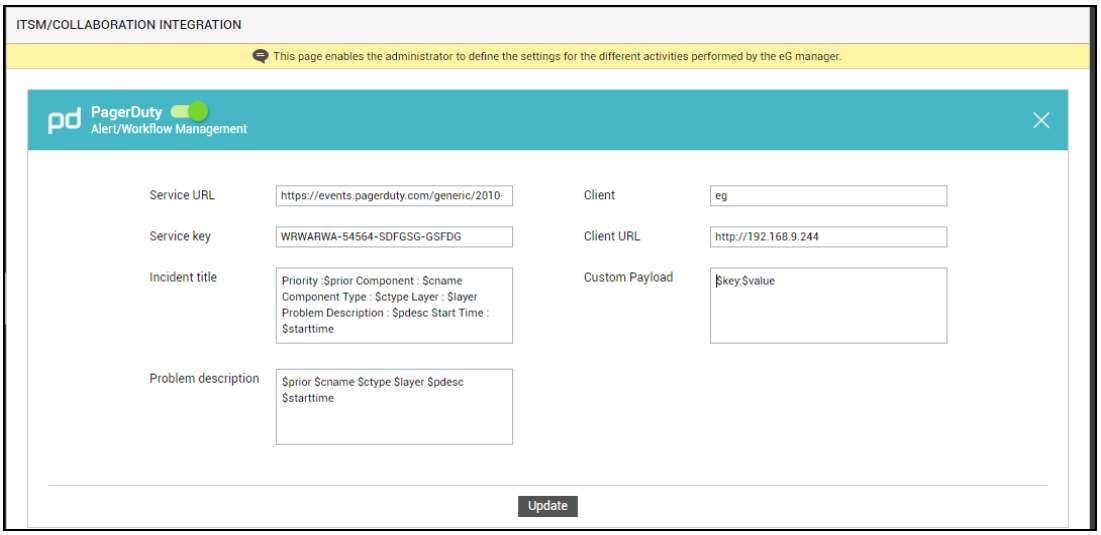

Figure 5.10: Configuring integration with PagerDuty

5. To enable integration with PagerDuty, first slide the **PagerDuty** slider in Figure 5.10 to the right.
6. Then, specify the following in Figure 2:

   - **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.
   - **Incident title**: Specify 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 key, 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 key $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 keys**.

     The other keys 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

- **Problem description**: Specify the format in which the problem description should be displayed in the trouble tickets generated by PagerDuty. The default problem description contains the following keys:

$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 keys or by changing the positions of a few keys. For instance, you may not want the problem component type to be part of the problem description. In this case therefore, your **Problem description** 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**.
- **Custom Payload**: Use custom payload to customize the alert information you send to PagerDuty, so that it includes additional static information.

Typically, the details of an eG alert are sent as a JSON file to the configured URL. Every piece of information contained within an eG alert - eg., priority, component name, component type etc. - is represented in the JSON file as a $key:$value pair, where 'key' denotes the alert field, and 'value' denotes the actual value of that field at run time. The 'key' is configured based on

the keys supported by the PagerDuty REST API. For instance, if the REST API represents alarm priorities using the key 'prior', then the same key will be used in the JSON file for denoting alarm priorities. Accordingly, the entry for alarm priority in the JSON file will be $prior:$value. The $value will be Critical, Major, Minor, or Normal, depending upon the actual priority of the alarm being sent.

- First, check whether the PagerDuty REST API supports a 'key' that can be used for capturing the 'source' of alerts/incidents. If no such key exists, then you cannot proceed with the Custom Payload configuration. On the other hand, if such a key is available, then proceed to replace the $key in your Custom Payload specification, with that key value. For the purpose of our example, let us assume that the REST API supports the key named 'source'. In this case therefore, substitute '$key' with 'source'.
- Then, proceed to explicitly specify the FQDN of your eG manager in the place of $value. This is because, you can use the Custom Payload configuration to add only 'static' information - i.e., information that you explicitly configure, and hence will never change. In the case of our example therefore, the $value will be egmanager.innovations.com.
- The complete **Custom Payload** specification will now be: *source:egmanager.innovations.com*

If you want eG incidents routed to PagerDuty to include additional information, then you can define a **Custom Payload** for that information as a *$key:$value* pair. For example, say, you want incidents to indicate the FQDN of the eG manager that generated the incidents. Say that the FQDN of your eG manager is *egmanager.innovations.com*. To include this information in PagerDuty incidents, do the following:

7. Finally, click the **Update** button in 5.5.

5.6 Integrating with HipChat

HipChat is a web service for internal private online chat and instant messaging. As well as one-on-one and group/topic chat, it also features cloud-based file storage, video calling, searchable message-history and inline-image viewing.

eG integration with HipChat ensures that eG alerts are automatically routed to a configured chat room, so that they can be circulated as messages amongst all users in that chat room. For instance, if help desk personnel are part of a single chat room on HipChat, you can have eG Enterprise integrate with that chat room. This way, help desk personnel are instantly notified of problem conditions, and can effectively work together for a speedy resolution.

The steps in this regard are as follows:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.11 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.11, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

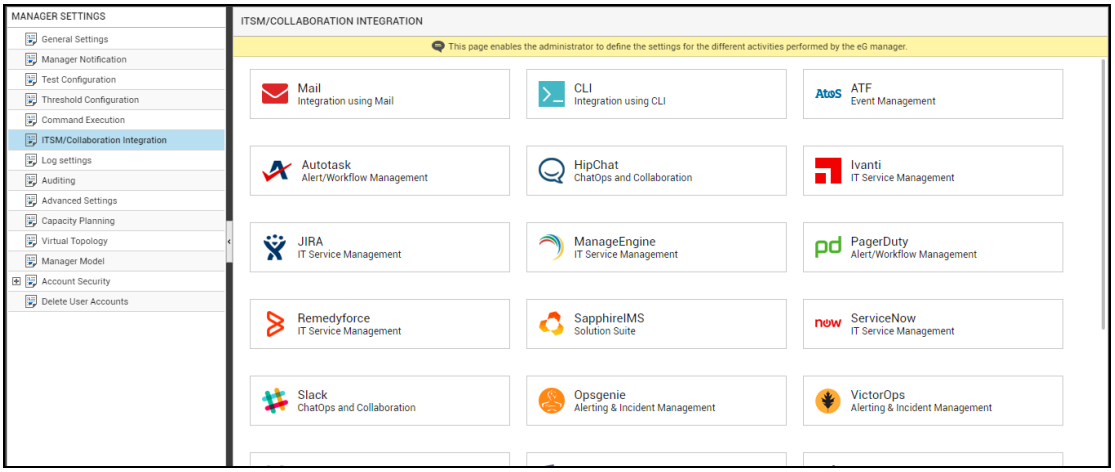

Figure 5.11: Viewing the ITSM/Collaboration tool options

4. Now, click on the **HipChat**option in the right panel (see Figure 5.11). A **HiptChat** section will now appear in the right panel (see Figure 5.12).

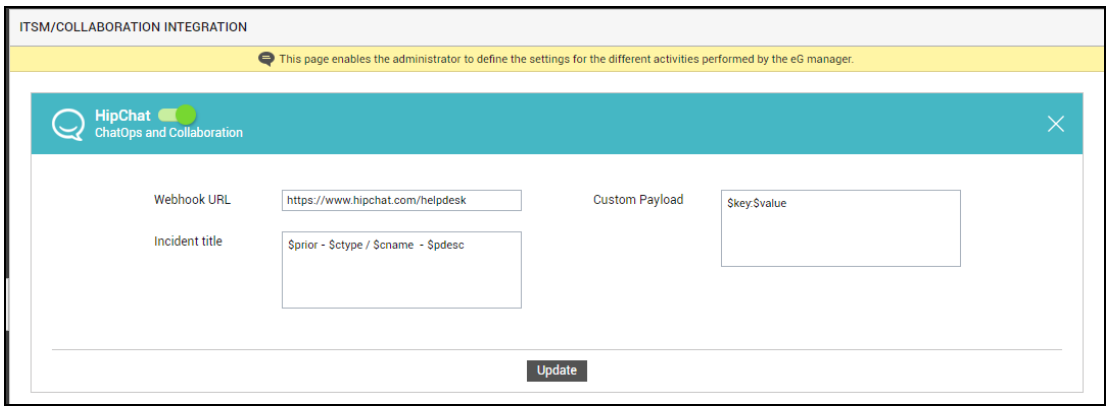

Figure 5.12: Configuring integration with HipChat

5. To enable integration with HipChat, first slide the **HipChat** slider in Figure 5.12 to the right.
6. Then, specify the following in Figure 5.12:

- **Webhook URL**: Webhooks are used by eG Enterprise to send event notifications to HipChat via REST API. A Webhook is typically associated with a specific chat room only. Therefore, indicate the chat room to which the eG alerts are to be transmitted by specifying that chat room's Webhook URL here.
- **Incident title**: Specify the format in which the title of the trouble ticket is to be displayed in HipChat. The default format is as follows:

```
$prior - $ctype / $cname  - $pdesc
```

The 'dollared' ($) text in the format above is a key, the value of which varies at run time, depending upon the information contained in the eG alarms. For example, in the default format above, $prior is a key that represents the alarms priority, and changes according to the priority of the actual alarm that is sent by the eG manager to HipChat. You are advised against changing any of the keys, as these are the keys that the HipChat REST API supports.

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
$pdesc	Will display a brief problem description

- **Custom payload**: Use custom payload to customize the alert information you send to HipChat, so that it includes additional static information.

Typically, the details of an eG alert are sent as a JSON file to the configured Webhook URL. Every piece of information contained within an eG alert - eg., priority, component name, component type etc. - is represented in the JSON file as a $key:$value pair, where 'key' denotes the alert field, and 'value' denotes the actual value of that field at run time. The 'key' is configured based on what the HipChat REST API supports. For instance, if the REST API represents alarm priorities using the key 'prior', then the same key will be used in the JSON file for denoting alarm priorities. Accordingly, the entry for alarm priority in the JSON file will be $prior:$value. The $value will be Critical, Major, Minor, or Normal, depending upon the actual priority of the alarm being sent.

If you want eG incidents routed to HipChat to include additional information, then you can define a Custom Payload for that information as a $key:$value pair. For example, say, you want incidents to indicate the FQDN of the eG manager that generated the incidents. Say that the FQDN of your eG manager is egmanager.innovations.com. To include this information in HipChat incidents, do the following:

- First, check whether the HipChat REST API supports a 'key' that can be used for capturing the 'source' of alerts/incidents. If no such key exists, then you cannot proceed with the Custom Payload configuration. On the other hand, if such a key is available, then proceed to replace the $key in your Custom Payload specification, with that key value. For the purpose of our example, let us assume that the REST API supports the key named 'source'. In this case therefore, substitute '$key' with 'source'.
- Then, proceed to explicitly specify the FQDN of your eG manager in the place of $value. This is because, you can use the Custom Payload configuration to add only 'static' information - i.e., information that you explicitly configure, and hence will never change. In the case of our example therefore, the $value will be egmanager.innovations.com.
- The complete Custom Payload specification will now be: *source:egmanager.innovations.com*

7. Finally, click the **Update** button in 5.6.

5.7 Integrating with Slack

Slack is a digital workspace that powers your organization - all the pieces and the people - so you can get things done.

A team is a group of people that use Slack to communicate. Larger organizations - possibly comprised of many locations, people, and sub-groups - may have multiple interconnected Slack workspaces. The Slack workspace is comprised of channels. Channels hold conversations between team members. They can be organized around anything - departments, projects, or even office locations - and you can create as many as you need.

eG integration with Slack ensures that eG alerts are automatically routed to a configured channel, so that they can be circulated amongst all team members using that channel. For instance, if help desk personnel are part of a team, then that team can be associated with a channel. You can have eG Enterprise integrate with that channel. This way, help desk personnel are instantly notified of problem conditions, and can effectively work together for a speedy resolution.

The steps in this regard are as follows:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.13 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure

5.13, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

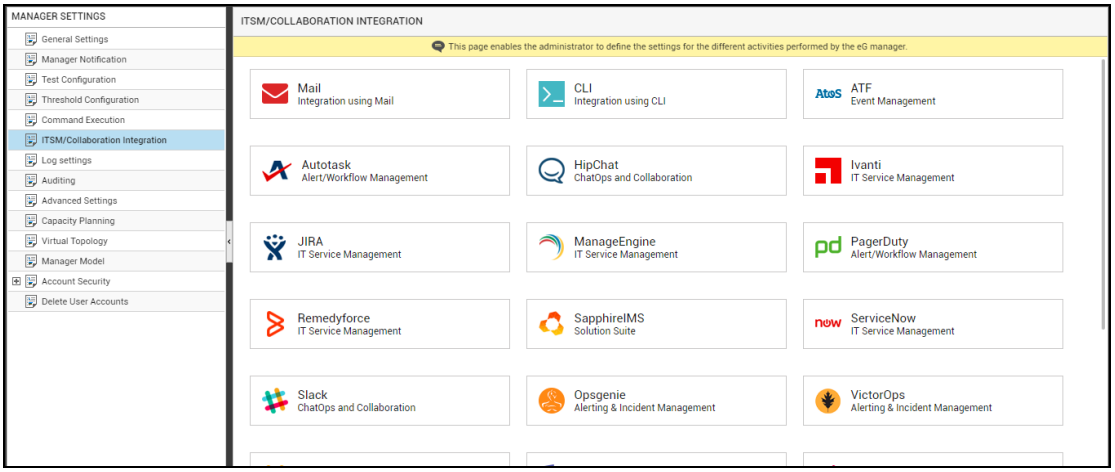

Figure 5.13: Viewing the ITSM/Collaboration tool options

4. Now, click on the **Slack**option in the right panel (see Figure 5.13). A **Slack** section will now appear in the right panel (see Figure 5.14).

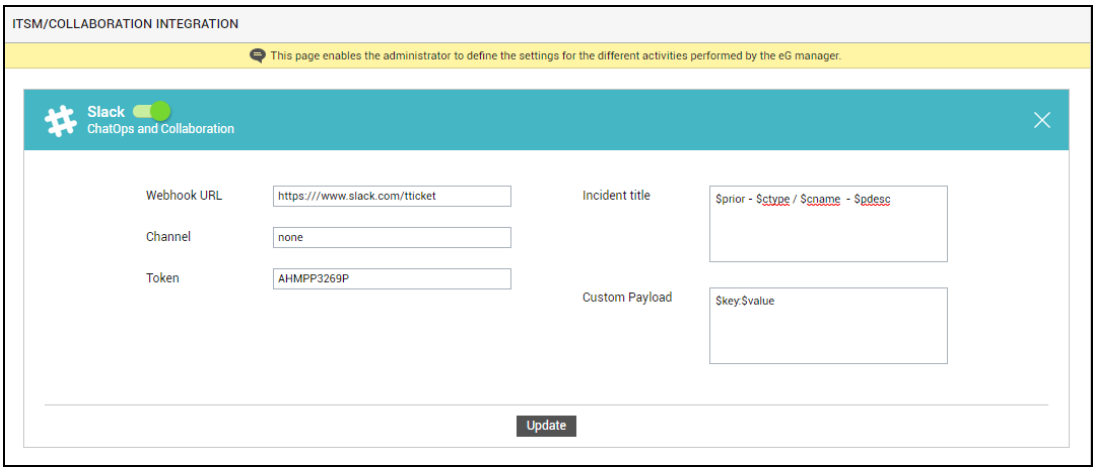

Figure 5.14: Configuring integration with Slack

5. To enable integration with Slack, first slide the **Slack** slider in Figure 5.14 to the right.
6. Then, specify the following in Figure 5.14:

- **Webhook URL**: Incoming Webhooks are a simple way to post messages from external sources into a Slack channel. eG Enterprise uses these Webhooks to send event notifications

to a specific Slack channel. Against Webhook URL, specify the Slack channel URL to which the eG alerts are to be transmitted.

- **Channel**: Specify the name of the Slack channel to which the eG alerts are to be sent.
- **Token**: Specify the token required to access the Slack **Channel** configured above.
- **Incident title**: Specify the format in which the title of the trouble ticket is to be displayed in Slack. The default format is as follows:

```
$prior - $ctype / $cname  - $pdesc
```

The 'dollared' ($) text in the format above is a key, the value of which varies at run time, depending upon the information contained in the eG alarms. For example, in the default format above, $prior is a key that represents the alarms priority, and changes according to the priority of the actual alarm that is sent by the eG manager to Slack. You are advised against changing any of the key names.

The other keys 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
$pdesc	Will display a brief problem description

- **Custom Payload**: Use custom payload to customize the alert information you send to Slack, so that it includes additional static information.

  Typically, the details of an eG alert are sent as a JSON file to the configured URL. Every piece of information contained within an eG alert - eg., priority, component name, component type etc. - is represented in the JSON file as a $key:$value pair, where 'key' denotes the alert field, and 'value' denotes the actual value of that field at run time. The 'key' is configured based on the keys supported by the Slack REST API. For instance, if the REST API represents alarm priorities using the key 'prior', then the same key will be used in the JSON file for denoting alarm priorities. Accordingly, the entry for alarm priority in the JSON file will be $prior:$value. The $value will be Critical, Major, Minor, or Normal, depending upon the actual priority of the alarm being sent.

  - First, check whether the Slack REST API supports a 'key' that can be used for capturing the 'source' of alerts/incidents. If no such key exists, then you cannot proceed with the Custom Payload configuration. On the other hand, if such a key is available, then proceed to replace the $key in your Custom Payload specification, with that key value. For the purpose of our example, let us assume that the REST API supports the key named 'source'. In this case

therefore, substitute '$key' with 'source'.

- Then, proceed to explicitly specify the FQDN of your eG manager in the place of $value. This is because, you can use the Custom Payload configuration to add only 'static' information - i.e., information that you explicitly configure, and hence will never change. In the case of our example therefore, the $value will be egmanager.innovations.com.
- The complete **Custom Payload** specification will now be: *source:egmanager.innovations.com*

If you want eG incidents routed to Slack to include additional information, then you can define a **Custom Payload** for that information as a *$key:$value* pair. For example, say, you want incidents to indicate the FQDN of the eG manager that generated the incidents. Say that the FQDN of your eG manager is *egmanager.innovations.com*. To include this information in Slack incidents, do the following:

7. Finally, click the **Update** button in Figure 2.

5.8 Integrating with JIRA

JIRA is a proprietary issue tracking product, developed by Atlassian. It provides bug tracking, issue tracking, and project management functions.

If the eG manager is integrated with JIRA, then eG alerts are automatically routed to JIRA, where they trigger the creation and assignment of trouble tickets to relevant maintenance personnel.

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

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.15 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.15, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

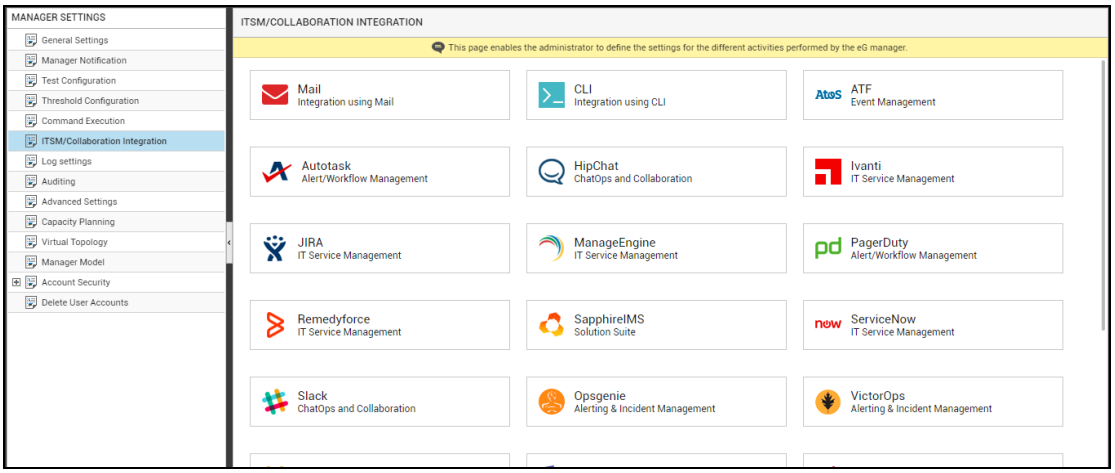

Figure 5.15: Viewing the ITSM/Collaboration tool options

4. Now, click on the **JIRA**option in the right panel (see Figure 5.15). A **JIRA** section will now appear in the right panel (see Figure 5.16).

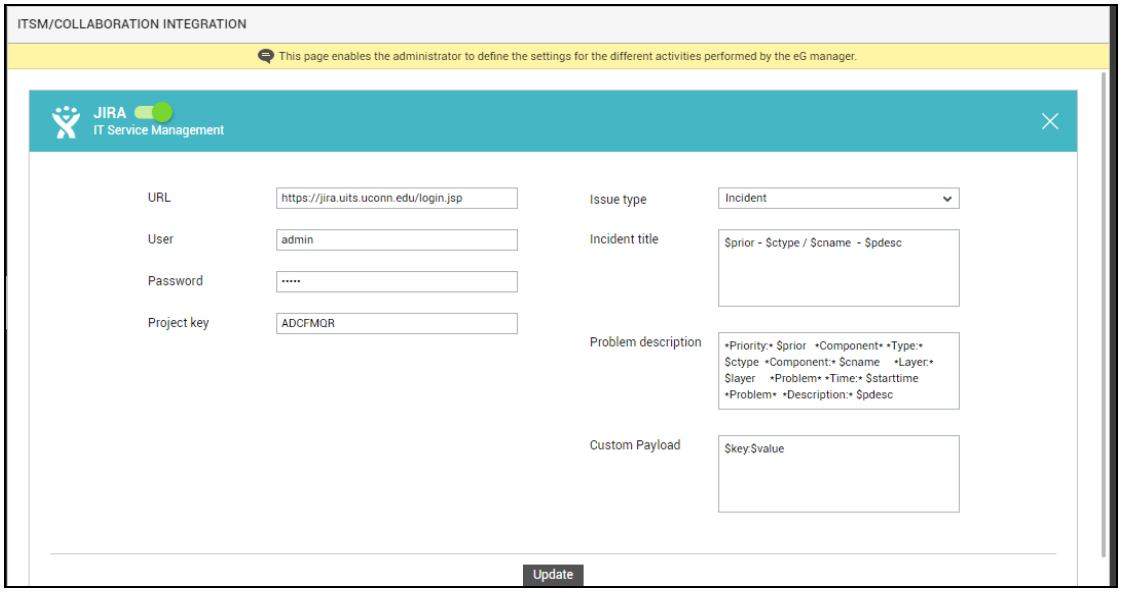

Figure 5.16: Configuring integration with JIRA

5. To enable integration with JIRA, first slide the **JIRA** slider in Figure 5.16 to the right.
6. Then, specify the following in Figure 5.16:

- **URL**: The web services URL using which the eG manager should connect to JIRA.
- **User name** and **Password**: Specify the credentials of a user who has the rights to access JIRA.
- **Project key**: A JIRA project is a collection of issues, and is defined according to an organisation's requirements. Each project is associated with a unique Project key. Specify the project key of the project with which the eG alerts are to be associated.
- **Issue type**: Indicate how you want to classify the eG alerts – as Incidents, Problems, or Service Requests. The default selection here is Incident.
- **Title format**: Specify the format in which the title of the trouble ticket is to be displayed in Slack. The default format is as follows:

```
$prior - $ctype / $cname  - $pdesc
```

The 'dollared' ($) text in the format above is a 'key', the value of which varies at run time, depending upon the information contained in the eG alarms. For example, in the default format above, $prior is a key that represents the alarms priority, and changes according to the priority of the actual alarm that is sent by the eG manager to JIRA. You are advised against changing any of the variable names.

The other keys 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
$pdesc	Will display a brief problem description

- **Description format**: Specify the format in which the problem description should be displayed in the trouble tickets generated by JIRA. The default problem description will be as follows:

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

As you can see, the problem description format comprises of two types of text strings – one that is enclosed within asterisks and one that follows a $ symbol. The text that is enclosed within asterisks represents a label that qualifies the value that follows. The 'dollared' ($) text on the other hand, is a key, the value of which varies at run time, depending upon the information contained in the eG alarms. The asterisk (*) itself represents an empty space. For example, in the default format above, Priority is a label that indicates that the information that follows it is the priority of the alarm. The key $prior that succeeds the label represents the alarm priority,

and changes according to the priority of the actual alarm that is sent by the eG manager to JIRA. While you can change the labels, you are advised against changing any of the keys.

The other keys 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

You can override this default format by removing one/more keys or by changing the positions of a few keys. 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.

```
*Priority:* $prior   *Component:* $cname     *Layer:* $layer     *Problem* *Time:*
$starttime   *Problem*  *Description:* $pdesc
```

- **Custom payload**: Use custom payload to customize the alert information you send to JIRA, so that it includes additional static information.

  Typically, the details of an eG alert are sent as a JSON file to the configured URL. Every piece of information contained within an eG alert - eg., priority, component name, component type etc. - is represented in the JSON file as a $key:$value pair, where 'key' denotes the alert field, and 'value' denotes the actual value of that field at run time. The 'key' is configured based on what the JIRA REST API supports. For instance, if the REST API represents alarm priorities using the key 'prior', then the same key will be used in the JSON file for denoting alarm priorities. Accordingly, the entry for alarm priority in the JSON file will be $prior:$value. The $value will be Critical, Major, Minor, or Normal, depending upon the actual priority of the alarm being sent.

  If you want eG incidents routed to JIRA to include additional information, then you can define a **Custom Payload** for that information as a $key:$value pair. For example, say, you want incidents to indicate the FQDN of the eG manager that generated the incidents. Say that the FQDN of your eG manager is egmanager.innovations.com. To include this information in JIRA incidents, do the following:

  - First, check whether the JIRA REST API supports a 'key' that can be used for capturing the 'source' of alerts/incidents. If no such key exists, then you cannot proceed with the **Custom Payload** configuration. On the other hand, if such a key is available, then proceed to replace

the $key in your **Custom Payload** specification, with that key value. For the purpose of our example, let us assume that the REST API supports the key named 'source'. In this case therefore, substitute '$key' with 'source'.

- Then, proceed to explicitly specify the FQDN of your eG manager in the place of $value. This is because, you can use the **Custom Payload** configuration to add only 'static' information - i.e., information that you explicitly configure, and hence will never change. In the case of our example therefore, the $value will be **egmanager.innovations.com**.
- The complete **Custom Payload** specification will now be: *source:egmanager.innovations.com*

7. Finally, click the **Update** button in 5.8.

5.9 Integration with ATF

With the Automated Test Framework (ATF), you create and run automated tests on your ServiceNow instance.

eG Enterprise integrates with ATF, so that eG alerts are automatically routed to ATF as and when they are generated. ATF then sends the alerts to ServiceNow Event Management for automatic creation/updation of trouble tickets.

To enable eG Enterprise to integrate with ATF, do the following:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.17 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.17, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

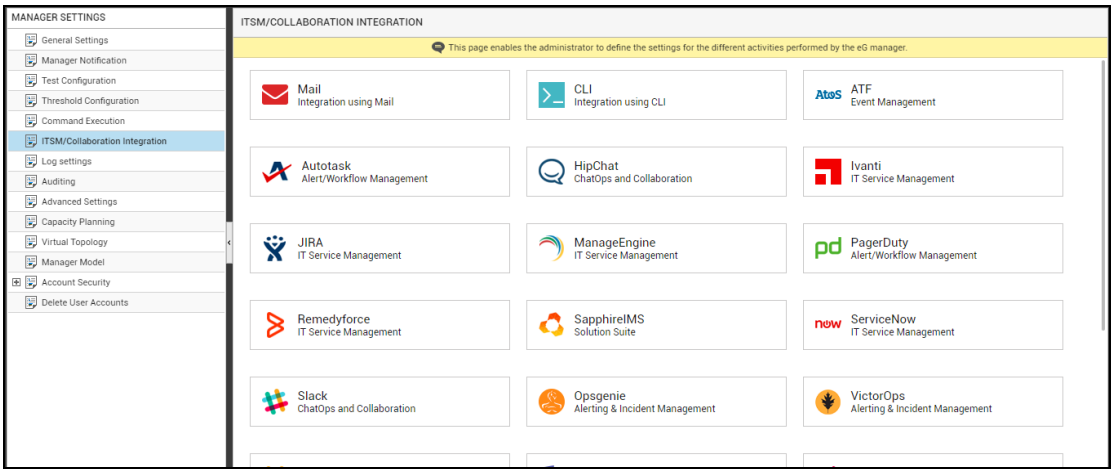

Figure 5.17: Viewing the ITSM/Collaboration tool options

4. Now, click on the **ATF**option in the right panel (see Figure 5.17). An **ATF** section will now appear in the right panel (see Figure 5.18).

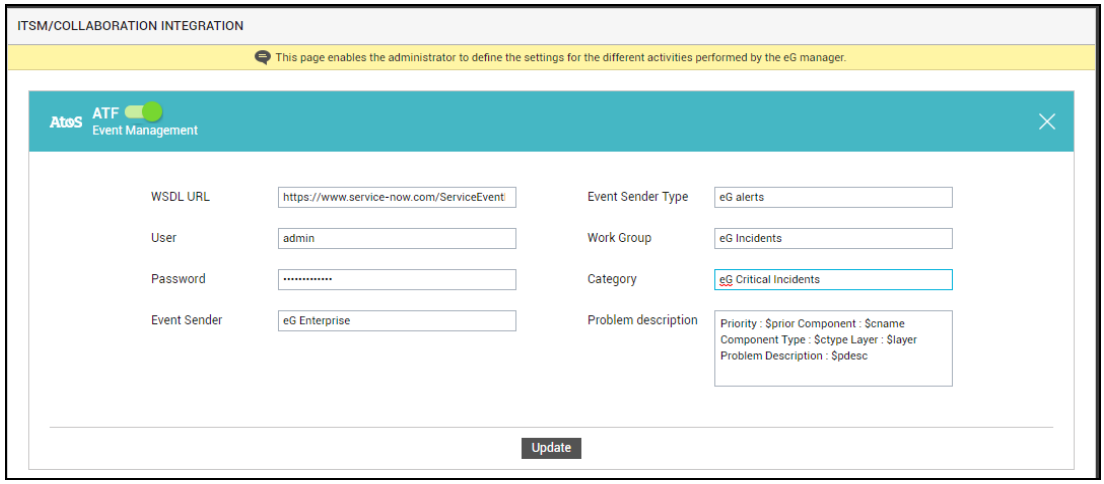

Figure 5.18: Configuring integration with ATF

5. To enable integration with ATF, first slide the **ATF** slider in Figure 5.18 to the right.
6. Then, specify the following in Figure 5.18:

- **WSDL URL**:Typically, all ServiceNow tables and import sets dynamically generate WSDL XML documents that describe its table schema and available operations. To integrate with ServiceNow via ATF, the eG manager needs access to these dynamic WSDL XML documents. For that, you need to configure the eG manager with a URL targeting the desired ServiceNow table with the WSDL parameter - eg., https://myinstance.service-

now.com/incident.do?WSDL. Specify this WSDL URL here.

- **User** and **Password**: ServiceNow requires that any request for a web service be accompanied by valid credentials of a user who is authorized to make that request. Specify the user name and password of such a user, here.
- **Event Sender**: Specify the IP/host name of the client that is sending the events into ServiceNow. In our case, this should be the IP/host name of the eG manager.
- **Event Sender Type**: Specify the event sender type.
- **Work Group**: A group is a set of users who share a common purpose. You can, if required, create a special work group in ServiceNow, which will be responsible for resolving eG incidents. If you create such a group, then specify the name of that group here.
- **Category**: Assigning incident tickets to categories allows for easy classification of incidents. If you want, you can create a new category for eG incidents in ServiceNow, and specify the name of that category here.
- **Problem description**: Next, using the **Problem description** text box, specify the format in which eG Enterprise should describe a problem when sending problem details into ServiceNow via ATF. 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 key, 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 key $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 keys**.

The other keys 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 will convert the alarm into a trouble ticket with the following problem description:

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

7. Finally, click the **Update** button in 5.9.

5.10 Integration with Ivanti Service Manager

Ivanti Service Manager, powered by HEAT, is a cloud-optimized ITSM solution, which automates workflows, eliminates manual processes, and improves business security and efficiency. Using the Service Manager, you can perform IT help desk / support ticketing or more advanced ITIL service management processes.

eG Enterprise integrates with Service Manager, so that eG alerts are automatically routed to Ivanti, resulting in the automatic creation/updation of trouble tickets.

To integrate eG Enterprise with the Ivanti Service Manager, do the following:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.19 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.19, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

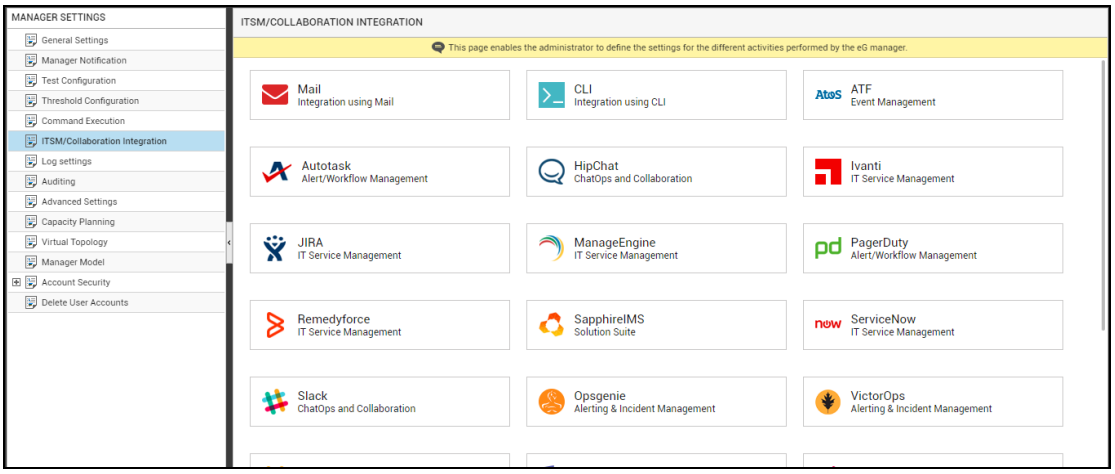

Figure 5.19: Viewing the ITSM/Collaboration tool options

4. Now, click on the **Ivanti**option in the right panel (see Figure 5.19). An **Ivanti** section will now appear in the right panel (see Figure 5.20).

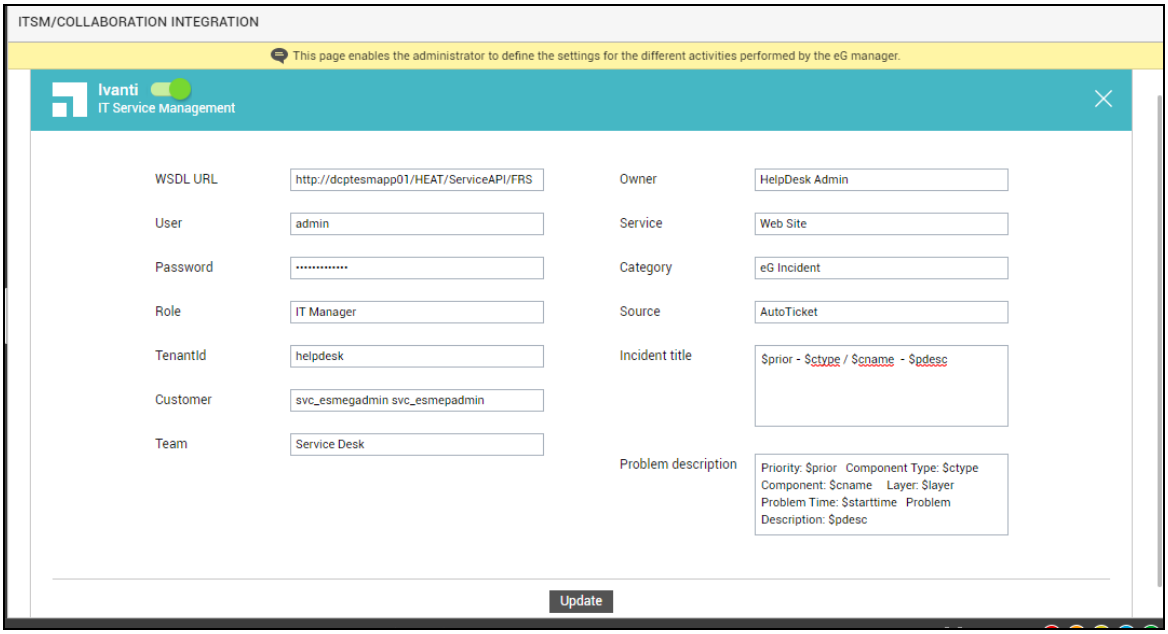

Figure 5.20: Configuring integration with Ivanti Service Manager

5. To enable integration with Ivanti Service Manager, first slide the **Ivanti** slider in Figure 5.20 to the right.
6. Then, specify the following in Figure 5.20:

- **WSDL URL**: Specify the Web Services Description Language (WSDL) URL via which the eG manager should connect to Service Manager's web services interface.
- **User** and **Password**: Specify the credentials of a user with access to Service Manager's web services interface.
- **Role**: Service Manager uses roles to define responsibilities for the users as they work within the application. A role consists of device- or function-specific application access to various workspaces, business objects, and fields. The eG manager needs to be assigned a role that has permissions to create/update trouble tickets in Service Manager. Assign such a **Role** to the eG manager.
- **TenantID** : A Tenant is an Ivanti Service Manager system for a particular customer or application. Since the eG manager will be integrating with a particular customer's or application's Service Manager system, the corresponding tenant ID has to be specified here.

- **Customer**: Specify the name of the person who is reporting the incident. To enable help desk to quickly figure out that it was the eG manager that reported the incident, you may want to use a special customer name for all eG alerts - eg., eG Monitor.
- **Team**: Mention the team that will troubleshoot the incident. If there is a dedicated team for attending to all eG alerts, then specify the name of that team here.
- **Owner**: Specify the login ID of the Service Desk Analyst who is assigned to work on the incident. All trouble tickets auto-generated from eG alerts will be assigned to this analyst only.
- **Service**: Mention the service that has been affected by the incident.
- **Category**: Categorizing incidents allows the Service Manager to appropriately assign the incidents to specialists by matching categories with skills. For example, an incident with a category of desktop hardware could be assigned to the desktop hardware group. This means that you can group all eG alerts into a specialized category, so they can be collectively assigned to an expert in troubleshooting eG alerts. If such a special category exists, then specify its name against **Category**.
- **Source**: Specify how the incident was submitted. Typically, this can be configured with values such as email, phone, etc. For eG alerts, the source is set by default, and it is *AutoTicket*.
- **Incident title**: Specify the format in which the title of the trouble ticket is to be displayed in Service Manager. The default format is as follows:

```
$prior - $ctype / $cname  - $pdesc
```

The 'dollared' ($) text in the format above is a key, the value of which varies at run time, depending upon the information contained in the eG alarms. For example, in the default format above, $prior is a key that represents the alarms priority, and changes according to the priority of the actual alarm that is sent by the eG manager to Service Manager. You are advised against changing any of the key names.

The other keys 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
$pdesc	Will display a brief problem description

- **Problem description**: Against **Problem description**, specify the format in which the problem description should appear in the trouble ticket that is auto-created for eG alerts, in the Service Manager. The default format is as follows:

```
Priority: $prior Component Type: $ctype Component: $cname Layer: $layer Problem
Time: $starttime 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 key, 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 key $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 the Service Manager. **While you can change the labels, you are advised against changing any of the key names.**

The other keys 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
$starttime	Will display the problem start time
$pdesc	Will display a brief problem description

7. Finally, click the **Update** button in Figure 5.20 to save the changes.

5.11 Integration with Moogsoft

Moogsoft AIOps is the pioneering AI platform for IT operations, powered by purpose-built machine learning algorithms. Moogsoft helps to improve the detection and remediation of incidents, ensuring continuous service delivery for business.

eG Enterprise integrates with Moogsoft, so that problem events detected by eG can be automatically sent to Moogsoft. Moogsoft then employs machine-learning and AI to analyze, correlate, deduplicate, and virtually share event information with all those responsible for resolving them, so as to slash event MTTR.

To integrate eG Enterprise with Moogsoft, do the following:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.21 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.21, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

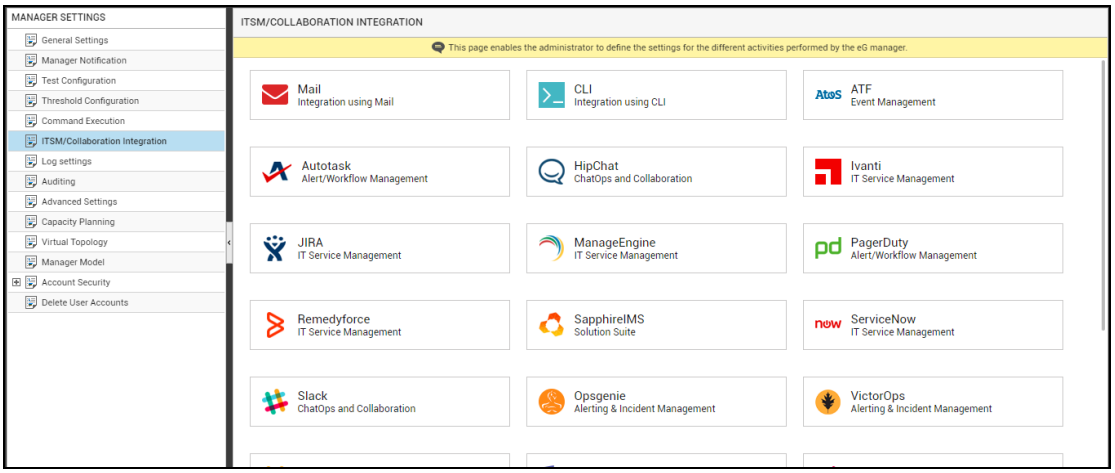

Figure 5.21: Viewing the ITSM/Collaboration tool options

4. Now, click on the **Moogsoft**option in the right panel (see Figure 5.21). A **Moogsoft** section will now appear in the right panel (see Figure 5.22).

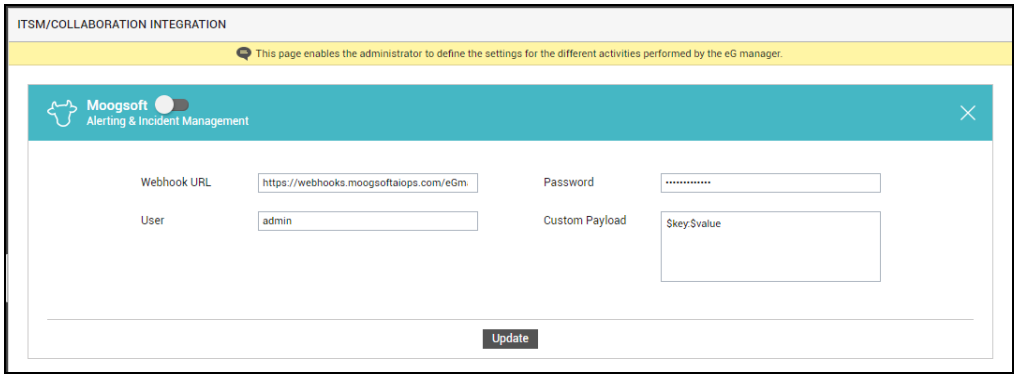

Figure 5.22: Configuring integration with Moogsoft

5. To enable integration with Moogsoft, first slide the **Moogsoft** slider in Figure 5.22 to the right.
6. Then, specify the following in Figure 5.22:

- **Webhook URL**: eG Enterprise uses webhook integration to route eG alerts to Moogsoft. This integration allows the eG manager to POST JSON payloads containing event information to the webhook URL of Moogsoft AIOps. To enable this communication, first specify the webhook URL of Moogsoft AIOps here.
- **User** and **Password**: Here, specify the credentials of a valid, basic authentication user, with rights to hit the configured webhook URL and post event details.

- **Custom payload**: Use custom payload to customize the alert information you send to Moogsoft, so that it includes additional static information.

  Typically, the details of an eG alert are sent as a JSON file to the configured REST URL. Every piece of information contained within an eG alert - eg., priority, component name, component type etc. - is represented in the JSON file as a $key:$value pair, where 'key' denotes the alert field, and 'value' denotes the actual value of that field at run time. The 'key' is configured based on what the Moogsoft LAM (Link Access Module) supports. For instance, if the LAM represents alarm priorities using the key 'prior', then the same key will be used in the JSON file for denoting alarm priorities. Accordingly, the entry for alarm priority in the JSON file will be $prior:$value. The $value will be Critical, Major, Minor, or Normal, depending upon the actual priority of the alarm being sent.

  If you want eG incidents routed to Moogsoft to include additional information, then you can define a **Custom Payload** for that information as a $key:$value pair. For example, say, you want incidents to indicate the FQDN of the eG manager that generated the incidents. Say that the FQDN of your eG manager is egmanager.innovations.com. To include this information in alerts sent to Moogsoft, do the following:

  - First, check whether the Moogsoft LAM supports a 'key' that can be used for capturing the 'source' of alerts/incidents. If no such key exists, then you cannot proceed with the Custom Payload configuration. On the other hand, if such a key is available, then proceed to replace the $key in your Custom Payload specification, with that key value. For the purpose of our example, let us assume that the supports the key named 'source'. In this case therefore, substitute '$key' with 'source'.
  - Then, proceed to explicitly specify the FQDN of your eG manager in the place of $value. This is because, you can use the Custom Payload configuration to add only 'static' information - i.e., information that you explicitly configure, and hence will never change. In the case of our example therefore, the $value will be egmanager.innovations.com.
  - The complete Custom Payload specification will now be: *source:egmanager.innovations.com*

7. Finally, click the **Update** button in Figure 5.22 to save the changes.

5.12 Integration with ConnectWise

ConnectWise is a business process automation platform that allows your business to sell, service and support technology more efficiently and in a more streamlined way.

ConnectWise has a CRM, ticketing system, help desk, and tools for project management, billing, and procurement.

eG Enterprise integrates with the ConnectWise ticketing and help desk system, so that eG alerts can be automatically forwarded to that system, thus automating the creation and updation of trouble tickets.

To integrate eG Enterprise with ConnectWise, do the following:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.23 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.23, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

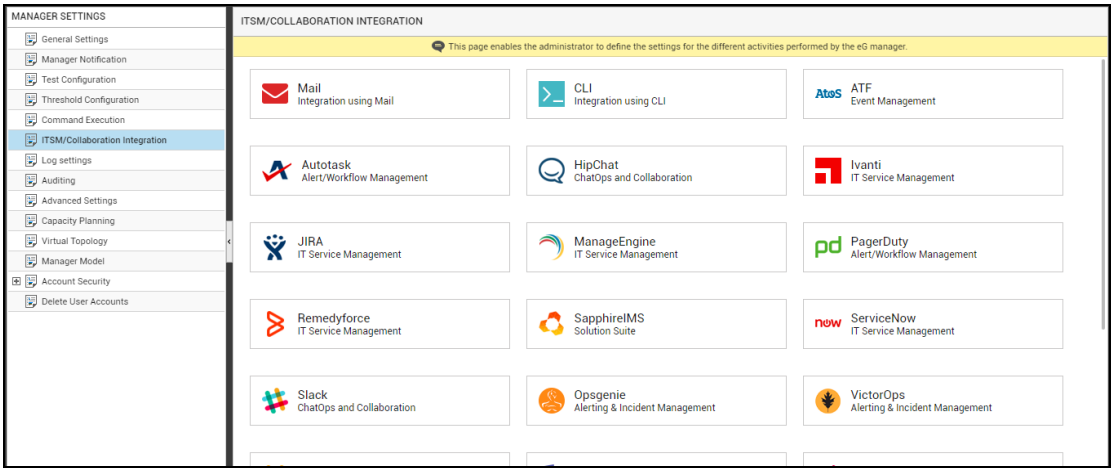

Figure 5.23: Viewing the ITSM/Collaboration tool options

4. Now, click on the **ConnectWise** option in the right panel (see Figure 5.23). A **ConnectWise** section will now appear in the right panel (see Figure 5.24).

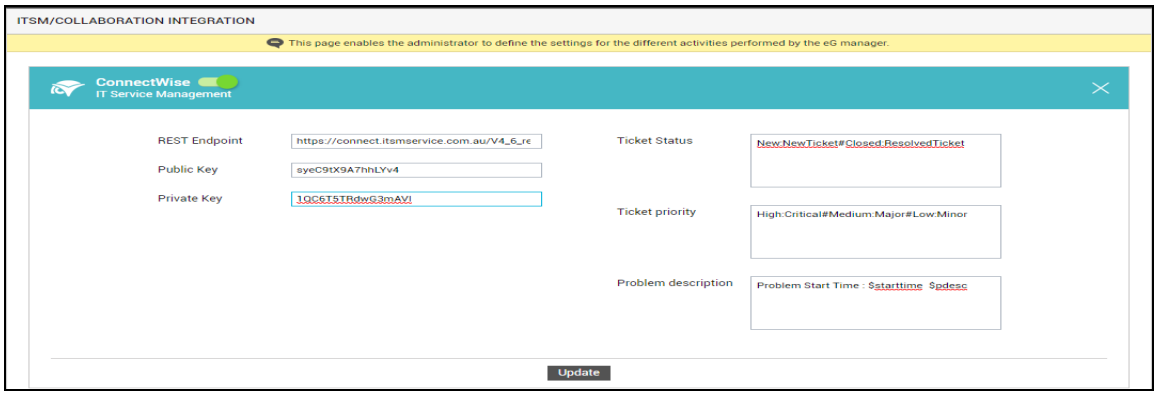

Figure 5.24: Configuring integration with ConnectWise

5. To enable integration with ConnectWise, first slide the **ConnectWise** slider in Figure 5.24 to the right.
6. Then, specify the following in Figure 5.24:

- **REST Endpoint**: eG Enterprise uses the ConnectWise REST API to integrate with it. One of the key requirements for this integration, is the REST endpoint URL. eG alerts are POSTed as JSON payloads to the ConnectWise endpoint URL via HTTP/HTTPS. Specify this URL here.
- **Public key** and **Private key**: Another key requirement for integrating eG Enterprise with ConnectWise are API keys. API keys are public and private keys that are generated by ConnectWise Manage for an integration account. In ConnectWise, you first need to create API keys exclusively for the eG manager integration. Make a note of these keys and then specify them against Public key and Private key here.
- **Ticket Status**: A ticket status can typically be New or Resolved. Using the Ticket Status text box, indicate the status strings ConnectWise uses to denote a new ticket and a resolved ticket. For that, you need to replace the '$Name' parameter that corresponds to NewTicket and ResolvedTicket in the Ticket Status specification, with the relevant status strings used by ConnectWise. For instance, if ConnectWise uses the status string 'New' for a new ticket, and 'Closed' for a resolved ticket, then your Ticket Status specification will be:

```
New:NewTicket#Closed:ResolvedTicket
```

- **Ticket priority**: eG Enterprise supports three alert/ticket priorities - Critical, Major, and Minor. Using the Ticket priority specification, indicate the ConnectWise-equivalent for each of these priorities - i.e., specify how ConnectWise represents Critical, Major, and Minor priorities in its trouble tickets. For this, you need to replace the $Name parameter corresponding to each eG

alert priority in the Ticket priority specification, with the relevant priority strings used by ConnectWise. For instance, if in a ConnectWise trouble ticket, the priority Critical is represented as High, Major is represented as Medium, and Minor is represented as Low, your Ticket priority specification should be:

```
High:Critical#Medium:Major#Low:Minor
```

- **Problem description**: Against **Problem description**, specify the format in which the problem description should appear in the ConnectWise trouble tickets that correspond to eG alerts. The default format is as follows:

```
Problem Start Time : $startTime $pdesc
```

The text preceding the ':' (colon) in the format above indicates what information follows. The 'dollared' ($) text that follows the ':' (colon) is a key, the value of which varies at run time, depending upon the eG alarms. For example, in the default format above, Problem Start Time is a label that indicates that the information that follows the ':' is the start time of the problem. The key $startTime that succeeds the ':' represents the actual start time of the problem that is sent by the eG manager to Opsgenie. **While you can change the labels, you are advised against changing any of the key names.**

The other key that is part of the default format is as follows:

$pdesc	Will display a brief problem description

7. Finally, click the **Update** button in Figure 5.24 to save the changes.

5.13 Integration with MS Teams

Microsoft Teams is a unified communication and collaboration platform that combines persistent workplace chat, video meetings, file storage, and application integration.

eG Enterprise integrates with a team channel, so that eG alerts are automatically routed to that channel. This way, users connected to that channel will be notified of anomalies and will be able to track them to closure.

To integrate eG Enterprise with MS Teams, do the following:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.25 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.25, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration

tools that eG Enterprise can integrate with will be listed in the right panel.

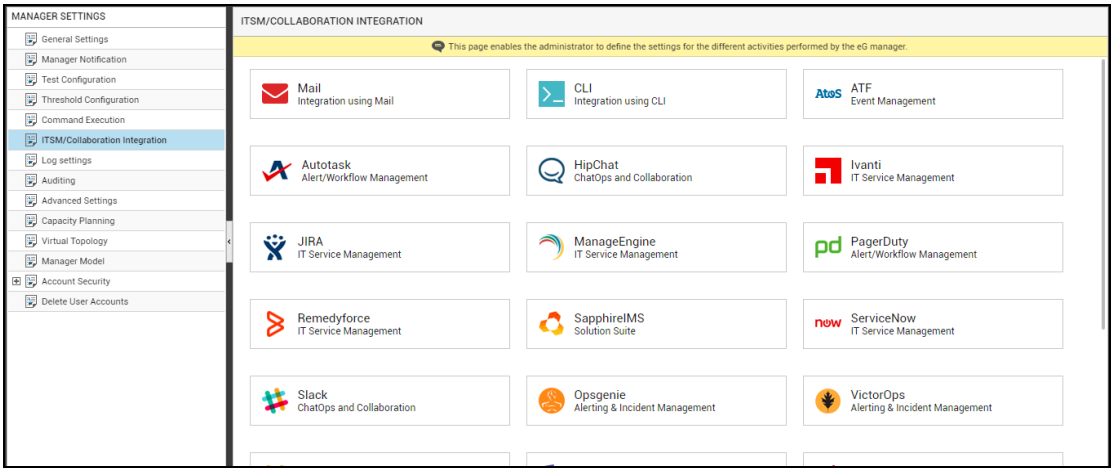

Figure 5.25: Viewing the ITSM/Collaboration tool options

4. Now, click on the **MS Teams**option in the right panel (see Figure 5.25). An **MS Teams** section will now appear in the right panel (see Figure 5.26).

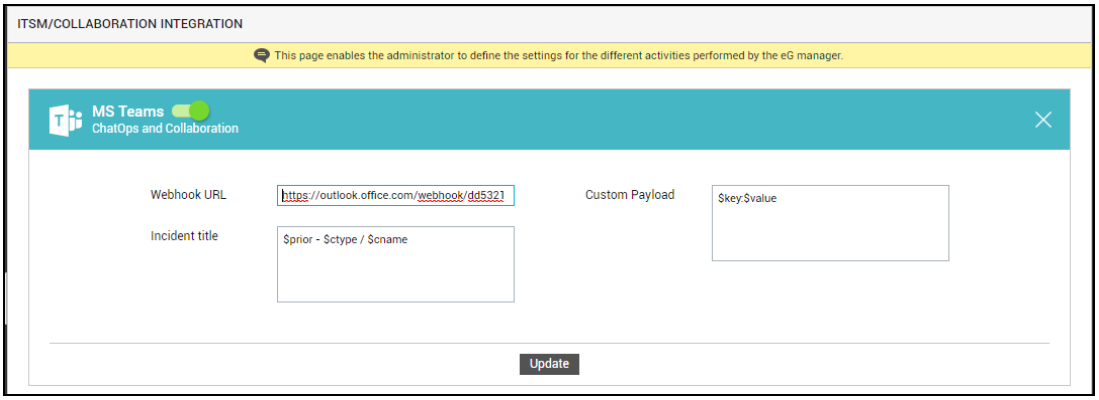

Figure 5.26: Configuring integration with MS Teams

5. To enable integration with MS Teams, first slide the **MS Teams** slider in Figure 5.26 to the right.
6. Then, specify the following in Figure 5.26:

   - **Webhook URL**: Specify the incoming webhook URL of the team channel with which the eG manager should integrate. Incoming webhooks are special type of Connector in Teams that provide a simple way for any external app to share content in team channels and are often used as tracking and notification tools. Teams will provide a unique URL to which the eG manager should send a JSON payload with the alert message it wants to POST. Refer to the

MS Teams documentation to know how to generate this URL. Once you obtain the URL, enter it here.

- **Incident title**: Specify the title format for all eG alerts that are routed to the team channel. The default format is as follows:

```
$prior - $ctype / $cname
```

The 'dollared' ($) text in the format above is a key, the value of which varies at run time, depending upon the information contained in the eG alarms. For example, in the default format above, $prior is a key that represents the alarms priority, and changes according to the priority of the actual alarm that is sent by the eG manager to VictorOps. **You are advised against changing any of the key names.**

The other keys 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

- **Custom payload**: Use custom payload to customize the alert information you send to a team channel, so that it includes additional static information.

Typically, the details of an eG alert are sent as a JSON file to the configured Webhook URL. Every piece of information contained within an eG alert - eg., priority, component name, component type etc. - is represented in the JSON file as a $key:$value pair, where 'key' denotes the alert field, and 'value' denotes the actual value of that field at run time. The 'key' is configured based on what the MS TeamsAPI supports. For instance, if the API represents alarm priorities using the key 'prior', then the same key will be used in the JSON file for denoting alarm priorities. Accordingly, the entry for alarm priority in the JSON file will be $prior:$value. The $value will be Critical, Major, Minor, or Normal, depending upon the actual priority of the alarm being sent.

If you want eG incidents routed to a team channel to include additional information, then you can define a **Custom Payload** for that information as a $key:$value pair. For example, say, you want incidents to indicate the FQDN of the eG manager that generated the incidents. Say that the FQDN of your eG manager is egmanager.innovations.com. To include this information in MS Teams incidents, do the following:

- First, check whether the MS Teams API supports a 'key' that can be used for capturing the 'source' of alerts/incidents. If no such key exists, then you cannot proceed with the Custom Payload configuration. On the other hand, if such a key is available, then proceed to replace

the $key in your Custom Payload specification, with that key value. For the purpose of our example, let us assume that the API supports the key named 'source'. In this case therefore, substitute '$key' with 'source'.

- Then, proceed to explicitly specify the FQDN of your eG manager in the place of $value. This is because, you can use the Custom Payload configuration to add only 'static' information - i.e., information that you explicitly configure, and hence will never change. In the case of our example therefore, the $value will be egmanager.innovations.com.
- The complete Custom Payload specification will now be: *source:egmanager.innovations.com*

7. Finally, click the **Update** button in Figure 5.26 to save the changes.

5.14 Integration with Opsgenie

Opsgenie is a modern incident management platform that ensures critical incidents are never missed, and actions are taken by the right people in the shortest possible time. Opsgenie receives alerts from your monitoring systems and custom applications and categorizes each alert based on importance and timing.

eG Enterprise integrates with Opsgenie, so Opsgenie automatically receives alerts raised by the eG manager and assigns them to the right expert for a speedy resolution.

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

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.27 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.27, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

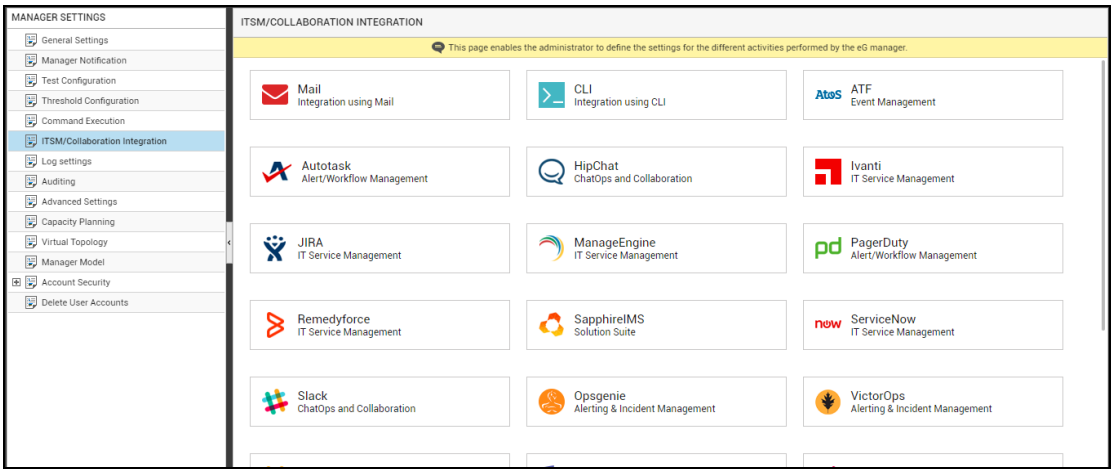

Figure 5.27: Viewing the ITSM/Collaboration tool options

4. Now, click on the **Opsgenie**option in the right panel (see Figure 5.27). An **Opsgenie** section will now appear in the right panel (see Figure 5.28).

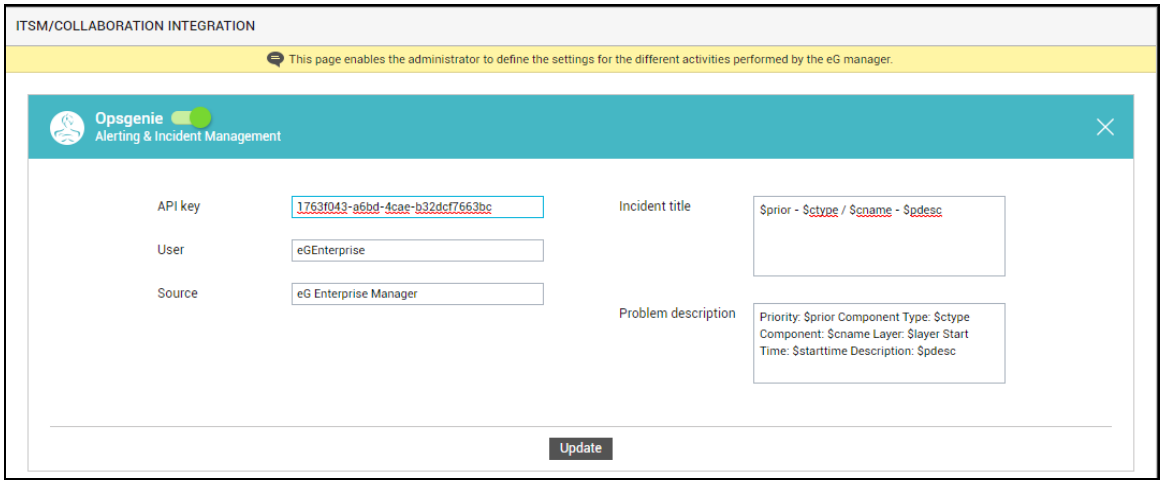

Figure 5.28: Configuring integration with Opsgenie

5. To enable integration with Opsgenie, first slide the **Opsgenie** slider in Figure 5.28 to the right.
6. Then, specify the following in Figure 5.28:

   - **API key**: eG Enterprise integrates with Opsgenie using the REST API of Opsgenie. Once the eG manager automatically routes eG alerts to Opsgenie, the REST API automatically creates corresponding alerts in Opsgenie and assigns them to the appropriate personnel. To enable the eG manager to make alert creation/updation requests to Opsgenie's REST API, you need

to configure the eG manager with the correct API key. This API key is auto-generated by Opsgenie as part of account-based configurations or for specific integrations. Specify the API key that is tied to the alert integration with eG Enterprise or with any account that is created in Opsgenie for eG Enterprise.

- **User**: Specify the name of the user who is reporting the problem. In our case, since it is the eG manager that is submitting an alert request with Opsgenie, you can tag all eG alerts going into Opsgenie with a common User name.
- **Source**: Indicate the source of the alert. As the eG manager is the source of alerts, you can specify the IP address of the eG manager here, or any text string that serves the same purpose.
- **Incident title**: Specify the title format for all eG alerts displayed in Opsgenie. The default format is as follows:

```
$prior - $ctype / $cname  - $pdesc
```

The 'dollared' ($) text in the format above is a key, the value of which varies at run time, depending upon the information contained in the eG alarms. For example, in the default format above, $prior is a key that represents the alarms priority, and changes according to the priority of the actual alarm that is sent by the eG manager to Opsgenie. **You are advised against changing any of the key names.**

The other keys 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
$pdesc	Will display a brief problem description

- **Problem description**: Against **Problem description**, specify the format in which the problem description should appear in the Opsgenie alerts that correspond to eG alerts. The default format is as follows:

```
Priority: $prior Component Type: $ctype Component: $cname Layer: $layer Start Time:
$starttime 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 key, 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 key $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 Opsgenie. **While you can change the labels, you are advised against changing any of the key names.**

The other keys 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
$starttime	Will display the problem start time
$pdesc	Will display a brief problem description

7. Finally, click the **Update** button in Figure 5.28 to save the changes.

5.15 Integration-with-SapphireIMS

SapphireIMS Service Desk is an ITIL 2011 certified, enterprise grade, comprehensive IT Service Management Suite. Using this solution, you can log incidents via multiple channels, categorize them, and have them automatically triaged and assigned to appropriate technicians as tickets.

eG Enterprise integrates with SapphireIMS Service Desk, so eG alerts can be automatically routed to Service Desk as and when they are raised by the eG manager. Then, using its Web Service API, Service Desk automatically converts the eG alerts into trouble tickets, and efficiently manages them with the help of its built-in workflow.

To integrate eG Enterprise with SapphireIMS Service Desk, do the following:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.29 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.29, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

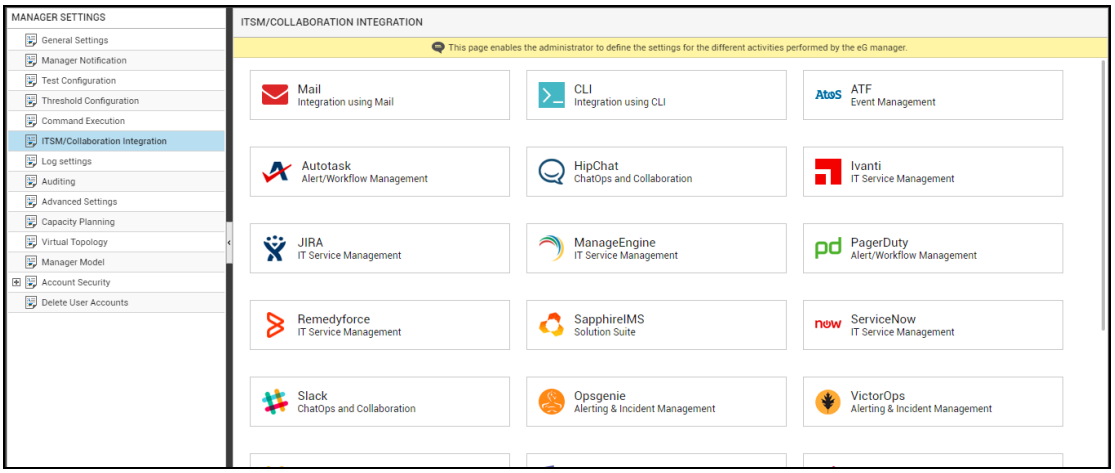

Figure 5.29: Viewing the ITSM/Collaboration tool options

4. Now, click on the **SapphireIMS** option in the right panel (see Figure 5.29). A **SapphireIMS** section will now appear in the right panel (see Figure 5.30).

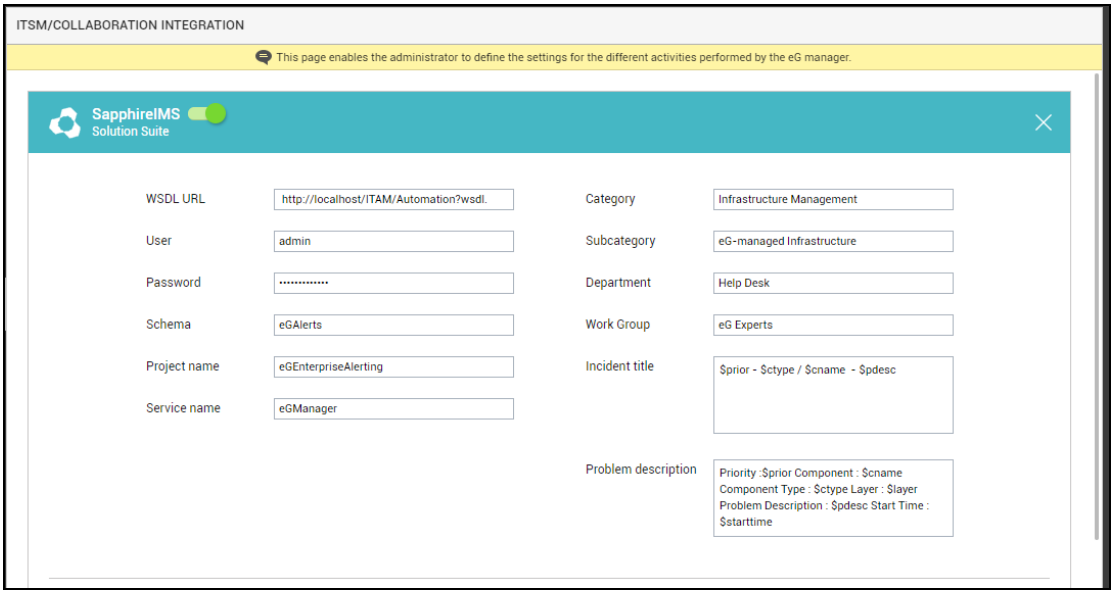

Figure 5.30: Configuring integration with SapphireIMS

5. To enable integration with SapphireIMS Service Desk, first slide the **SapphireIMS** slider in Figure 5.30 to the right.

6. Then, specify the following in Figure 5.30:

- **WSDL URL**: Specify the Web Services Description Language (WSDL) URL via which the eG manager should connect to service desk's web services API.
- **User** and **Password**: To create tickets in service desk, the eG manager requires the authentication key of a valid service desk user, with ticket creation rights. To obtain this authentication key, specify the credentials of that user here.
- **Schema**: Specify the database schema in which the ticket should be created. For Professional setup, schema name is 'ims'. For MSP setup, schema name should be the customer schema.
- **Project name**: Set the project to which the ticket applies. To obtain the project ID, specify the name of the project here.
- **Service name**: A service desk ticket has to be compulsorily mapped to a particular service. For example, for all hardware related issues, 'Desktop Management' service must be selected. Specify the service to which you want to map eG alerts.
- **Category**:A service desk record should be associated with a category to enable classification of the records. Category helps in assigning appropriate service desk technicians or users to work on the record. By grouping eG alerts into a single category, you can collectively assign them to an expert in troubleshooting eG alerts. Specify the name of the **Category** to which eG alerts belong. If the default categories cannot be used, you can add an exclusive category for eG alerts in service desk, and specify the name of that category here.
- **Subcategory**: Sub-Category is a sub set of the category. For e.g. System Administration category can have sub-categories like 'Hardware','Software' etc. Mention the sub-category for eG alerts here. If the default sub-categories cannot be used, you can add an exclusive sub-category for eG alerts in service desk, and specify the name of that sub-category here.
- **Department**: Specify the department of the submitter - i.e., the user who is reporting the incidents. Since in our case, the eG manager will be reporting incidents, you can create a new department in service desk for the sole purpose of the integration, and specify that department here.
- **Work Group**: Work group is useful for grouping services, categories, locations and roles. For example, only 'Application Service' related records can be moved to 'Application Group'. Using work groups, you can configure privileges to a group of people to work on a service desk record. In service desk, you can create a dedicated work group for eG alerts, and map it to specific services, categories, sub-categories, and user roles. You can then specify the name of this work group here. This way, you can have a select group of people troubleshooting eG alerts.

- **Incident title**: Specify the title format for all trouble tickets auto-generated for eG-reported incidents. The default title format is as follows:

```
$prior - $ctype / $cname  - $pdesc
```

The 'dollared' ($) text in the format above is a key, the value of which varies at run time, depending upon the information contained in the eG alarms. For example, in the default format above, $prior is a key that represents the alarms priority, and changes according to the priority of the actual alarm that is sent by the eG manager to Slack. You are advised against changing any of the key names.

The other keys 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
$pdesc	Will display a brief problem description

- **Problem description**: Against **Problem description**, specify the format in which the problem description should appear in the trouble ticket that is auto-created for eG alerts, in service desk. 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 key, 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 key $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 service desk. **While you can change the labels, you are advised against changing any of the key names.**

The other keys 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
$starttime	Will display the problem start time
$pdesc	Will display a brief problem description

7. Finally, click the **Update** button in Figure 5.30 to save the changes.

5.16 Integration with SNOW ITOM

ServiceNow® ITOM Enterprise delivers a comprehensive and integrated set of ITOM capabilities including infrastructure discovery, event management, automation/orchestration, operational intelligence, and many more. The ServiceNow Event Management solution in particular, consolidates, correlates, and analyzes data from all of your monitoring tools to deliver real-time information about the health of business services and IT infrastructure.

eG Enterprise integrates with ServiceNow Event Management, so that eG alerts can be automatically sent into ServiceNow for correlation and analysis.

To integrate eG Enterprise with ServiceNow ITOM, do the following:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.31 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.31, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

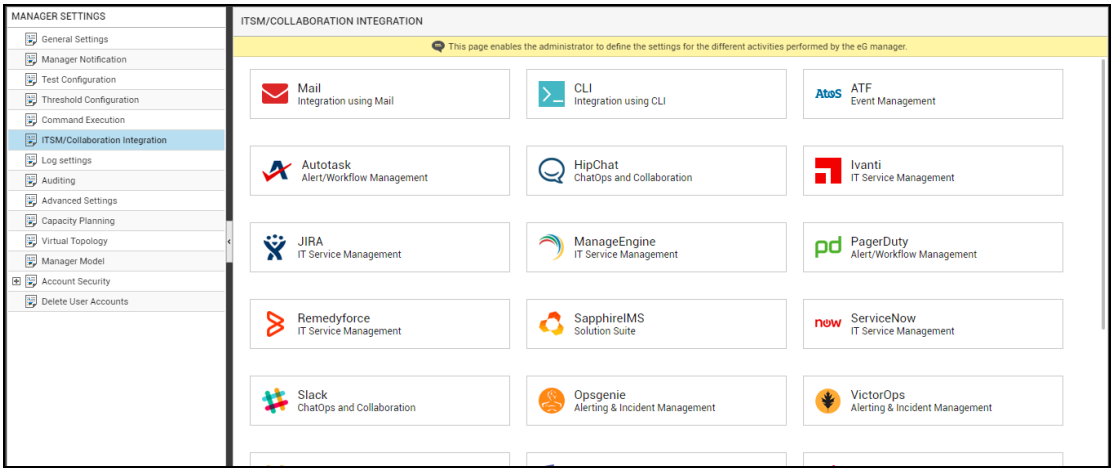

Figure 5.31: Viewing the ITSM/Collaboration tool options

4. Now, click on the **SNOW ITOM** option in the right panel (see Figure 5.31). A **SNOW ITOM** section will now appear in the right panel (see Figure 5.32).

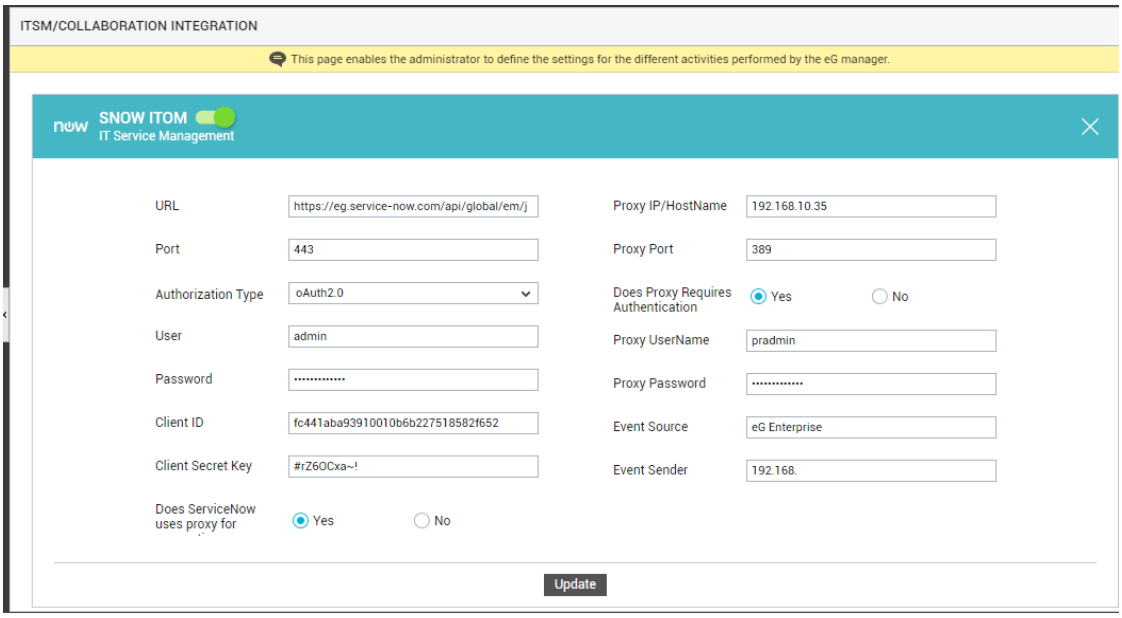

Figure 5.32: Configuring integration with SNOW ITOM

5. To enable integration with SNOW ITOM, first slide the **SNOW ITOM** slider in Figure 5.32 to the right.
6. Then, specify the following in Figure 5.32:

   - **URL**: eG Enterprise integrates with ServiceNow using its web service API. The eG manager POSTs eG alerts to the endpoint URL of the API as JSON payloads containing alert information. To enable the eG manager to connect to the API, you need to specify the endpoint URL here.
   - **Port**: The Port at which ServiceNow listens for problem information sent by the eG manager.
   - **Authorization Type**: The eG manager sends alarm information to ServiceNow as a web service request to the configured **URL**. Upon receipt of the request, ServiceNow will attempt to validate the source of the request using one of the following authentication methods:
     - Basic authentication
     - O Auth 2.0 authentication

     If ServiceNow enforces Basic Authentication, then select the **Basic** option from the **Authorization Type** drop-down. Where Basic Authentication is enforced, ServiceNow requires that web service requests be accompanied by the username and password of a user who has access to the ServiceNow instance. Accordingly, if **Basic** is set as the **Authorization**

Type, you need to provide the credentials of a user with the right to access ServiceNow in the **User** and **Password** text boxes.

On the other hand, if ServiceNow enforces the O Auth 2.0 authentication method, then select the **OAuth 2.0** option from the **Authorization Type** drop-down. O Auth 2.0 lets users access instance resources through external clients by obtaining a token rather than by entering credentials with each resource request. This means that where O Auth 2.0 is enforced, the eG manager needs to obtain an access token, so it can create/modify trouble tickets in ServiceNow. For this, the eG manager should first connect to the ServiceNow instance as a user who is authorized to request for an access token, and then submit web service requests as a valid 'Client'. This is why, if **OAuth 2.0** is set as the **Authorization Type**, you will have to specify the following:

- **User** and **Password**: The credentials of a user who is authorized to request for an access token
- **Client ID** and **Client Secret Key**: The **Client ID** is an auto-generated unique ID of the client application - i.e., in our case, the eG manager application - requesting the access token. The **Client Secret Key** is a shared secret string that the ServiceNow instance and the client applications - i.e., the eG manager - use to authorize communications with one another.

- **Does ServiceNow use proxy for connections**: If the eG manager needs to communicate with the ServiceNow instance via a Proxy server, then set this flag to **Yes**.
- **Proxy IP/Hostname** and **Proxy Port**: If the **Does ServiceNow use proxy flag** is set to **Yes**, then specify the IP/host name of the Proxy server and the port number at which the Proxy server listens in the respective text boxes.
- **Does Proxy require authentication**: This flag is applicable only if the **Does ServiceNow use proxy** flag is set to **Yes**. If so, then use this flag to indicate whether/not the Proxy server requires authentication. Set this flag to **Yes**, only if the Proxy server requires authentication.
- **Proxy UserName** and **Proxy Password**: If the **Does Proxy require authentication** flag is set to **Yes**, then provide the credentials of a valid Proxy user here.
- **Event Source**: Specify the tool that is sending events into ServiceNow. For the purpose of our integration, this can be set to *eG Enterprise*.
- **Event Sender**: Specify the IP/host name of the client that is sending the events into ServiceNow. In our case, this should be the IP/host name of the eG manager.

7. Finally, click the **Update** button in 5.16.

5.17 Integration with Zendesk

Zendesk is a customer support platform that lets you connect with customers on any channel. It boasts of an omnichannel ticketing system that helps collect all your customer support requests from any source and lets you manage them from one location.

eG Enterprise integrates with Zendesk's ticketing system, so that eG alerts are automatically fed into Zendesk as and when they are generated. This results in the automatic creation/updation of trouble tickets, the quick assignment of tickets to the right troubleshooting agents, and the speedy resolution of issues.

To integrate eG Enterprise with Zendesk, do the following:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.33 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.33, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

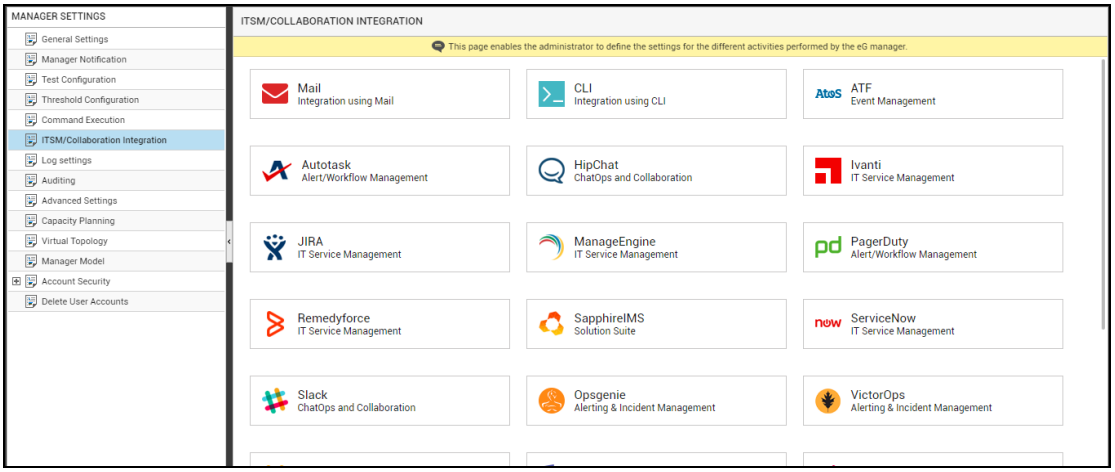

Figure 5.33: Viewing the ITSM/Collaboration tool options

4. Now, click on the **Zendesk**option in the right panel (see Figure 5.33). A **Zendesk** section will now appear in the right panel (see Figure 5.34).

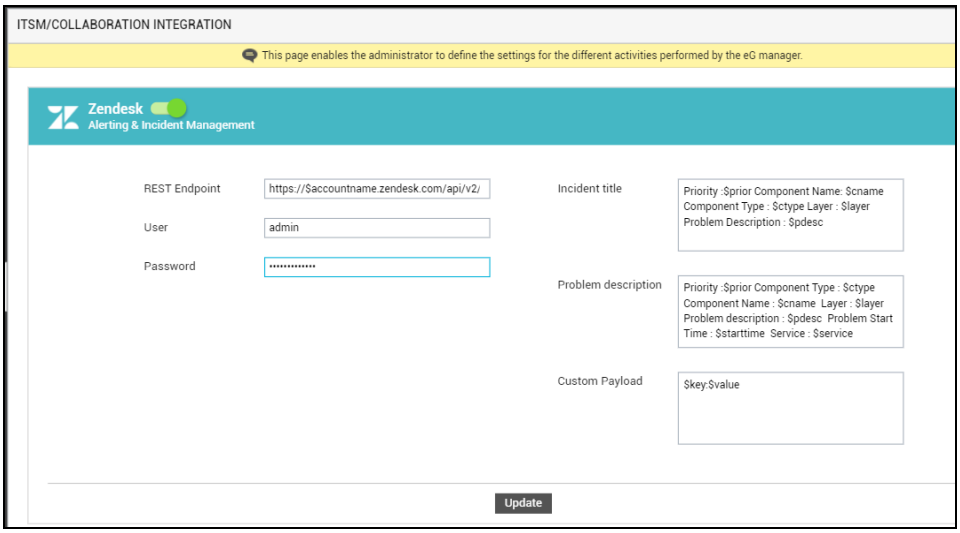

Figure 5.34: Configuring integration with Zendesk

5. To enable integration with Zendesk, first slide the **Zendesk** slider in Figure 5.34 to the right.
6. Then, specify the following in Figure 5.34:

- **URL**: eG Enterprise integrates with Zendesk using its REST API. Via HTTP/S, eG alerts are POSTed to the REST endpoint URL of Zendesk as JSON payloads containing problem information. To enable this communication, specify the REST endpoint URL here.
- **User** and **Password**: Only verified users can make requests to the Zendesk API. Therefore, to make a POST request to the API, the eG manager has to first authorize against the API using basic authentication. For this, it requires the privileges of a user with ticket creation rights. Specify the credentials of such a user against User and Password.
- **Incident title**: Specify the title format for all trouble tickets auto-generated for eG-reported incidents. The default title format is as follows:

```
Priority :$prior Component Name: $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 key, 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 key $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 Zendesk. **While you can change the labels, you are advised against changing any of the key names.**

The other keys that are part of the default title 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 name of the problematic layer
$pdesc	Will display a brief problem description

- **Problem description**: Against **Problem description**, specify the format in which the problem description should appear in the trouble ticket that is auto-created for eG alerts, in Zendesk. The default format is as follows:

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

The text preceding the ':' (colon) in the format above indicates what information follows. The 'dollared' ($) text that follows the ':' (colon) is a key, 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 key $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 service desk. **While you can change the labels, you are advised against changing any of the key names.**

The other keys 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
$starttime	Will display the problem start time
$pdesc	Will display a brief problem description
$service	Will display the name of the affected service

- **Custom payload**: Use custom payload to customize the alert information you send to Zendesk, so that it includes additional static information.

Typically, the details of an eG alert are sent as a JSON file to the configured URL. Every piece of information contained within an eG alert - eg., priority, component name, component type etc. - is represented in the JSON file as a $key:$value pair, where 'key' denotes the alert field,

and 'value' denotes the actual value of that field at run time. The 'key' is configured based on what the Zendesk REST API supports. For instance, if the REST API represents alarm priorities using the key 'prior', then the same key will be used in the JSON file for denoting alarm priorities. Accordingly, the entry for alarm priority in the JSON file will be $prior:$value. The $value will be Critical, Major, Minor, or Normal, depending upon the actual priority of the alarm being sent.

If you want eG incidents routed to VictorOps to include additional information, then you can define a **Custom Payload** for that information as a $key:$value pair. For example, say, you want incidents to indicate the FQDN of the eG manager that generated the incidents. Say that the FQDN of your eG manager is egmanager.innovations.com. To include this information in Zendesk tickets, do the following:

- First, check whether the Zendesk REST API supports a 'key' that can be used for capturing the 'source' of alerts/incidents. If no such key exists, then you cannot proceed with the Custom Payload configuration. On the other hand, if such a key is available, then proceed to replace the $key in your Custom Payload specification, with that key value. For the purpose of our example, let us assume that the REST API supports the key named 'source'. In this case therefore, substitute '$key' with 'source'.
- Then, proceed to explicitly specify the FQDN of your eG manager in the place of $value. This is because, you can use the Custom Payload configuration to add only 'static' information - i.e., information that you explicitly configure, and hence will never change. In the case of our example therefore, the $value will be egmanager.innovations.com.
- The complete Custom Payload specification will now be: *source:egmanager.innovations.com*

7. Finally, click the **Update** button in Figure 5.34 to save the changes.

5.18 Integration with VictorOps

VictorOps is a real-time incident management and collaboration platform for IT and DevOps teams.

By integrating eG Enterprise with VictorOps, you can have the performance alerts that eG generates sent into your VictorOps timeline and automatically trigger and resolve incidents.

To integrate eG Enterprise with VictorOps, do the following:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.

3. Figure 5.35 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.35, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

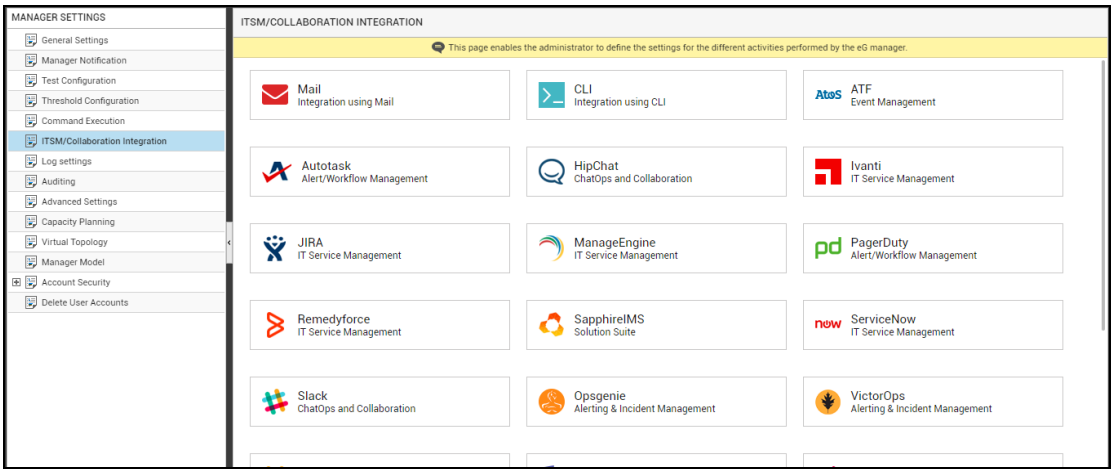

Figure 5.35: Viewing the ITSM/Collaboration tool options

4. Now, click on the **VictorOps**option in the right panel (see Figure 5.35). A **VictorOps** section will now appear in the right panel (see Figure 5.36).

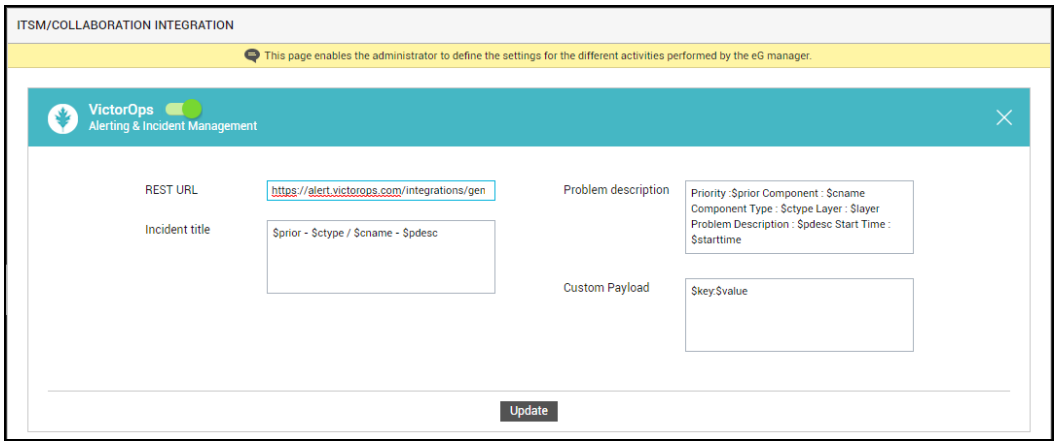

Figure 5.36: Configuring integration with VictorOps

5. To enable integration with VictorOps, first slide the **VictorOps** slider in Figure 5.36 to the right.
6. Then, specify the following in Figure 5.36:

- **REST URL**: eG Enterprise uses the REST API of VictorOps to integrate with it. Using the API, VictorOps converts eG Enterprise alerts into incidents, and automatically assigns these

incidents to the right person at the right time, thus ensuring the prompt redressal of performance issues. To enable eG Enterprise to connect to the REST API, you first need to enable the REST endpoint in VictorOps. Refer to the VictorOps documentation to know how to achieve this. Once the REST endpoint URL is enabled, VictorOps generates an URL, where alerts can be sent via an HTTP POST request. Specify this URL as the REST URL here.

- **Incident title**: Specify the title format for all eG alerts displayed in VictorOps as incidents. The default format is as follows:

```
$prior - $ctype / $cname  - $pdesc
```

The 'dollared' ($) text in the format above is a key, the value of which varies at run time, depending upon the information contained in the eG alarms. For example, in the default format above, $prior is a key that represents the alarms priority, and changes according to the priority of the actual alarm that is sent by the eG manager to VictorOps. **You are advised against changing any of the key names.**

The other keys 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
$pdesc	Will display a brief problem description

- **Problem description**: Against **Problem description**, specify the format in which the problem description should appear in the VictorOps incidents that correspond to eG alerts. 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 key, 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 key $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 VictorOps. **While you can change the labels, you are advised against changing any of the key names.**

The other keys 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
$starttime	Will display the problem start time
$pdesc	Will display a brief problem description

- **Custom payload**: Use custom payload to customize the alert information you send to VictorOps, so that it includes additional static information.

  Typically, the details of an eG alert are sent as a JSON file to the configured REST URL. Every piece of information contained within an eG alert - eg., priority, component name, component type etc. - is represented in the JSON file as a $key:$value pair, where 'key' denotes the alert field, and 'value' denotes the actual value of that field at run time. The 'key' is configured based on what the VictorOps REST API supports. For instance, if the REST API represents alarm priorities using the key 'prior', then the same key will be used in the JSON file for denoting alarm priorities. Accordingly, the entry for alarm priority in the JSON file will be $prior:$value. The $value will be Critical, Major, Minor, or Normal, depending upon the actual priority of the alarm being sent.

  If you want eG incidents routed to VictorOps to include additional information, then you can define a **Custom Payload** for that information as a $key:$value pair. For example, say, you want incidents to indicate the FQDN of the eG manager that generated the incidents. Say that the FQDN of your eG manager is egmanager.innovations.com. To include this information in VictorOps incidents, do the following:

  - First, check whether the VictorOps REST API supports a 'key' that can be used for capturing the 'source' of alerts/incidents. If no such key exists, then you cannot proceed with the Custom Payload configuration. On the other hand, if such a key is available, then proceed to replace the $key in your Custom Payload specification, with that key value. For the purpose of our example, let us assume that the REST API supports the key named 'source'. In this case therefore, substitute '$key' with 'source'.
  - Then, proceed to explicitly specify the FQDN of your eG manager in the place of $value. This is because, you can use the Custom Payload configuration to add only 'static' information - i.e., information that you explicitly configure, and hence will never change. In the case of our example therefore, the $value will be egmanager.innovations.com.
  - The complete Custom Payload specification will now be: *source:egmanager.innovations.com*

7. Finally, click the **Update** button in Figure 5.36 to save the changes.

5.19 Webhook Integration

A webhook is an HTTP callback which allows one application to provide other applications with real-time information. eG Enterprise is capable of posting event messages - i.e., eG alerts - to any third-party trouble ticketing system that is capable of accepting incoming webhooks.

To implement this integration, do the following:

1. Login to the eG administrative interface.
2. Select the **Manager** option from the **Settings** tile.
3. Figure 5.37 will then appear. From the **MANAGER SETTINGS** tree in the left panel of Figure 5.37, select the **ITSM/Collaboration Integration** node. The third-party ITSM/Collaboration tools that eG Enterprise can integrate with will be listed in the right panel.

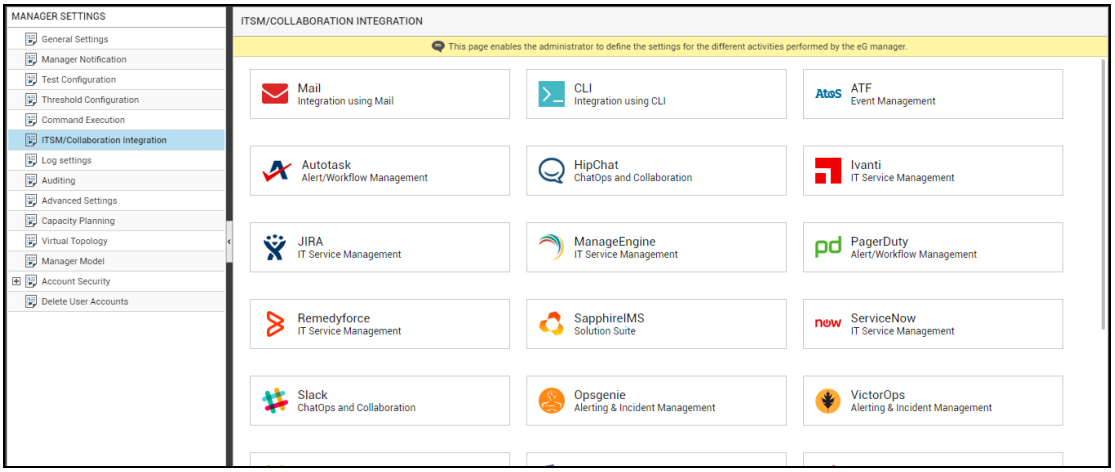

Figure 5.37: Viewing the ITSM/Collaboration tool options

4. Now, click on the **Webhook Integration**option in the right panel (see Figure 5.37). A **Webhook Integration** section will now appear in the right panel (see Figure 5.38).

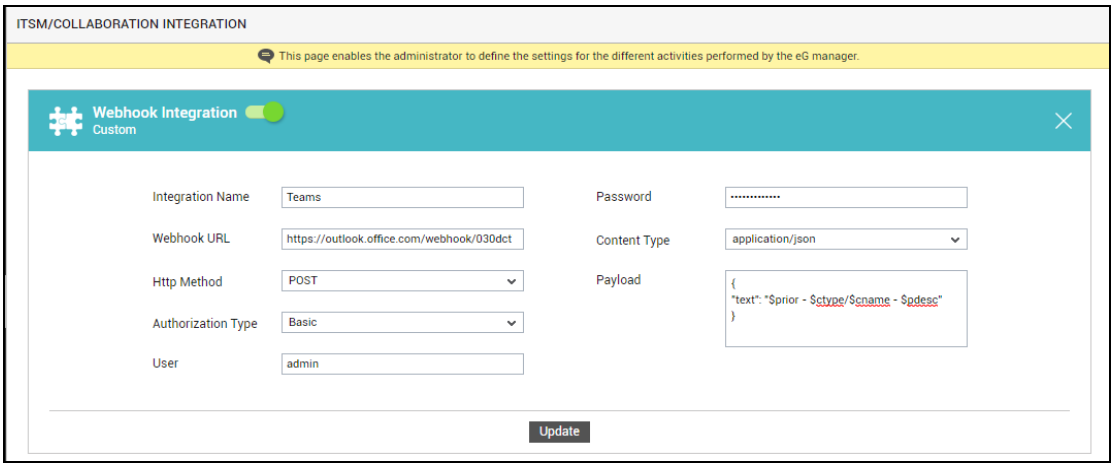

Figure 5.38: Configuring integration using Webhooks

5. To enable integration using webhooks, first slide the **Webhook Integration** slider in Figure 5.38 to the right.
6. Then, specify the following in Figure 5.38:

   - Integration Name: Specify the name of trouble ticketing system with which eG Enterprise should integrate.
   - **Webhook URL**: Here, specify the incoming webhook URL of the target trouble ticketing system. eG alerts will be transmitted to this URL only.
   - **HTTP Method**: Select the HTTP method that eG Enterprise uses to send event notifications into the trouble ticketing system. The options are GET, POST, and PUT.
   - **Authorization Type**: Some trouble ticketing systems require authorization from the external source to accept the incoming webhook. If the target of this integration requires such an authorization, then first select the type of authorization from this drop-down. The options are as follows:
     - Basic authentication
     - API Key

     If the target trouble ticketing system enforces Basic Authentication, then select the **Basic** option from the **Authorization Type** drop-down. Where Basic Authentication is enforced, the trouble ticketing system requires that webhooks be accompanied by the credentials of a valid user with the right to send webhooks. In this case therefore, provide the credentials of such a user in the **User** and **Password** text boxes.

Some other trouble ticketing systems may authenticate incoming webhooks using a unique API key. If the eG manager is being integrated with such a trouble ticketing system, then first generate an API key using the third-party system. Next, when configuring eG integration with that trouble ticketing system, set the **Authorization Type** to **API Key** and specify the correct **API key**.

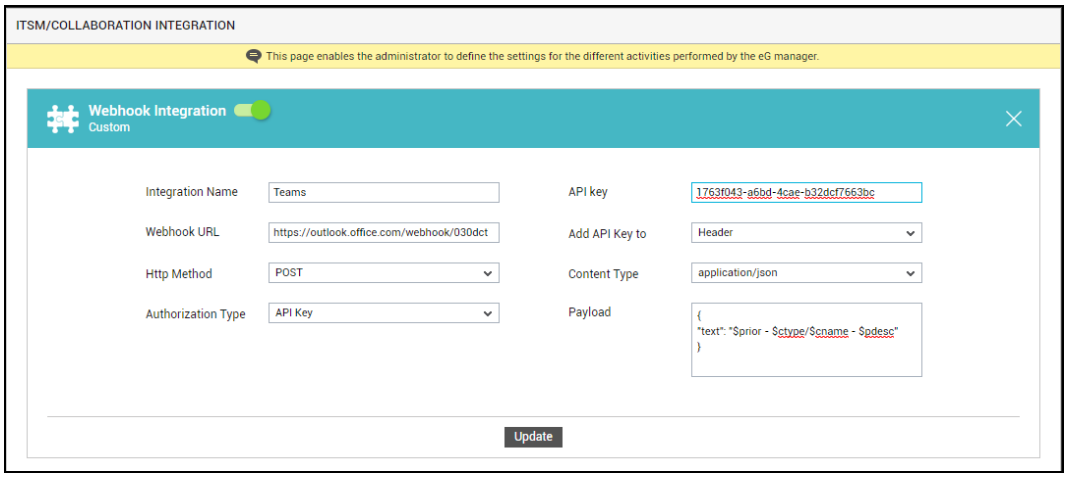

Figure 5.39: Setting Authorization Type to API Key

Then, by selecting the appropriate option from the **Add API Key** to drop-down, indicate where the API key needs to be inserted - in the HTTP request header? or in query params?

- **Content Type**: Then, indicate in what format eG alert information is to be sent to the third-party trouble ticketing system. Since the eG manager typically sends alarm information as JSON files, the application/json option is chosen by default here.
- **Payload**: Configure the event payload that is to be fed into the target trouble ticketing system. In other words, configure the type of information that each eG alert sent to the target system should contain. A sample payload specification is provided below:

```
{
"text": "$prior - $ctype/$cname - $pdesc"
}
```

The ‘dollared’ ($) text in the format above is a key, the value of which varies at run time, depending upon the information contained in the eG alarms. For example, in the default format above, $prior is a key that represents the alarms priority, and changes according to the priority of the actual alarm that is sent by the eG manager to the target system.

The other keys that are part of the sample 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
$pdesc	Will display a brief problem description

Before specifying the keys, you will have to check which keys are supported by the target trouble ticketing system. Only such keys should be configured in your payload.

7. Finally, click the **Update** button in Figure 5.38.

5.20 Adding Custom Fields to the Trouble Ticket Integration Page

Sometimes, you may want eG Enterprise to transmit more information to a TT system than what eG Enterprise's default monitoring and integration framework allows. For example, when integrating with Service Now, eG Enterprise by default sends the name and type of the problem component, the problem priority, the problem layer, and problem description as part of the eG alarm information. Service Now users may however want a Subcategory to also be set for eG alerts at the time of alarm transmission, so that, when browsing the eG alerts in the Service Now console, they can at-a-glance determine which area of work the issue pertains to. While a Subcategory field is available in Service Now, eG Enterprise's monitoring framework does not by default collect or report 'problem subcategory' as part of alarm information. To make sure that eG Enterprise sends this Subcategory information to Service Now along with the eG alerts, you first need to set a subcategory for the eG alerts in eG Enterprise. This can be achieved by adding **Subcategory** as a **Custom field** to the **Trouble Ticket Integration** page (in the eG admin interface) of Service Now.

Custom Fields enable eG Enterprise to capture and send problem information that is specific to a TT system.

This facility is currently available for the following TT systems only:

- Service Now
- Pager Duty
- HipChat
- Slack
- JIRA

The sub-sections below discuss how this can be achieved for **Service Now** and **JIRA** alone. The procedure for adding custom fields for other TT systems is similar to that of Service Now.

5.20.1 Adding Custom Fields for Service Now Integration

The key pre-requisite for adding a custom field for a specific TT system in eG Enterprise is that the field that you want to add should be supported by the third-party TT system at the time of addition.

Note:

The custom field that you want to add can either be available by default in the TT system or could have been added as a custom column to the TT system.

Assume that you want to add a field/parameter named '*Subcategory*' to the Trouble Ticket Integration page for Service Now in the eG admin interface. The first step towards this is to confirm that the '*Subcategory*' field is available in Service Now. Once this is confirmed, proceed as directed below:

1. Find the column name that corresponds to '*Subcategory*' (in our example) in Service Now. For this, follow the steps below:

   - Login to the Service Now console. Figure 5.40 will then appear. Follow the menu sequence Incident -> Create New, in the menu options available in the left panel of Figure 5.40.

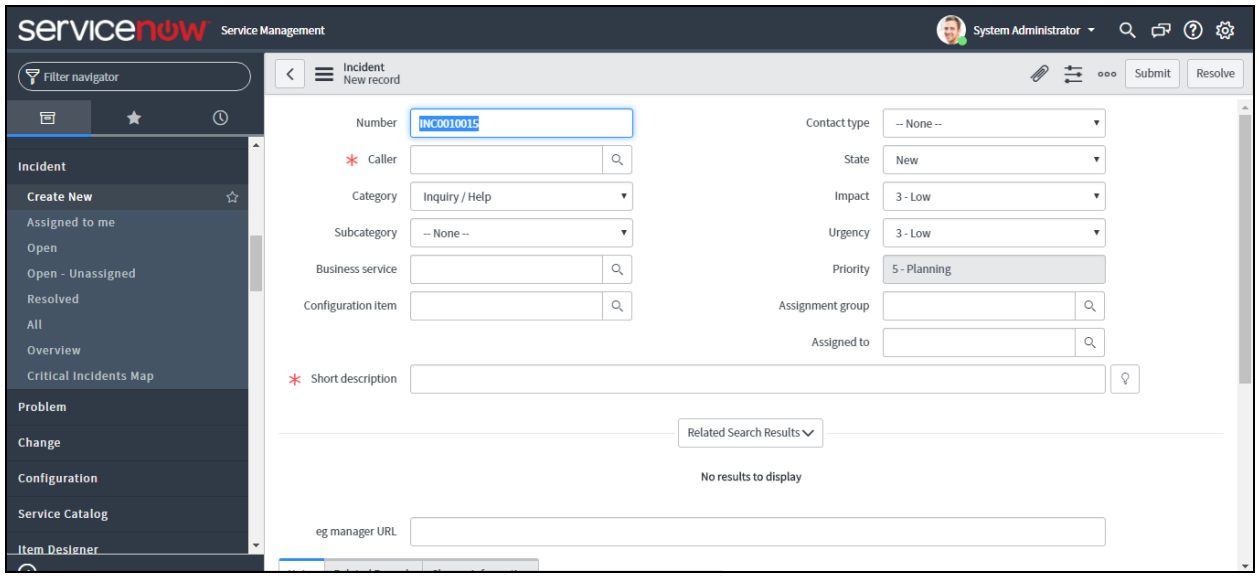

Figure 5.40: Creating a new incident

   - Clicking on the ☰ button at the top of the right panel in Figure 5.40 will bring up a shortcut menu as shown by Figure 5.41. From this menu, select the Configure -> Table option.

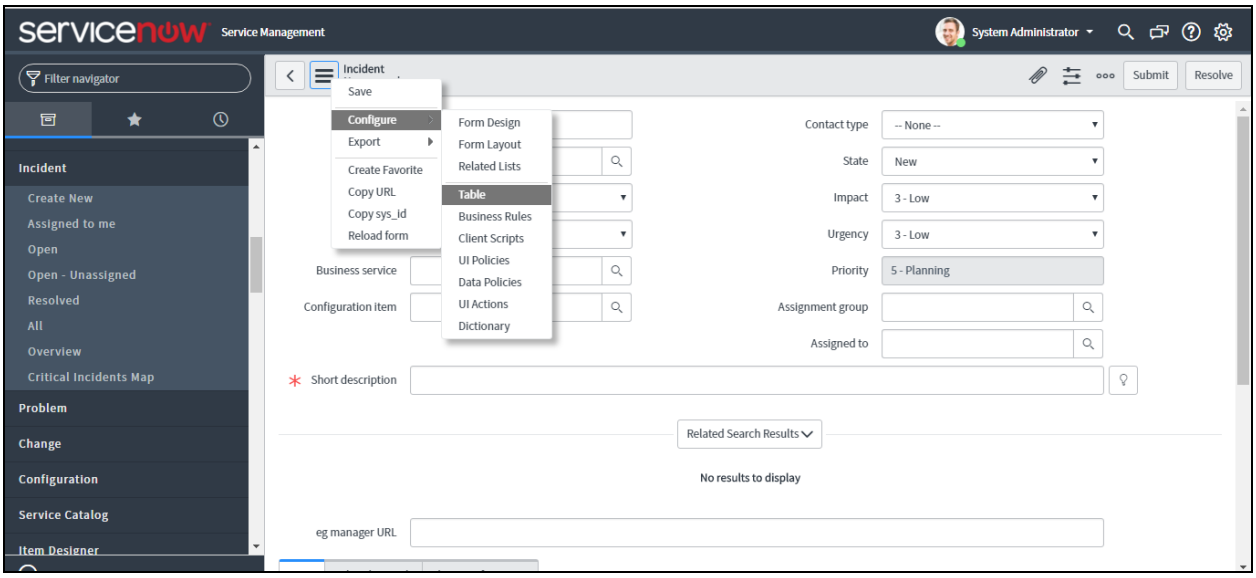

Figure 5.41: Selecting the Configure -> Table menu option

- Figure 5.42 will then appear. Use the **Search** option indicated by Figure 5.42 to search for the column named 'subcategory'. Once it is found, click on it as indicated by Figure 5.42.

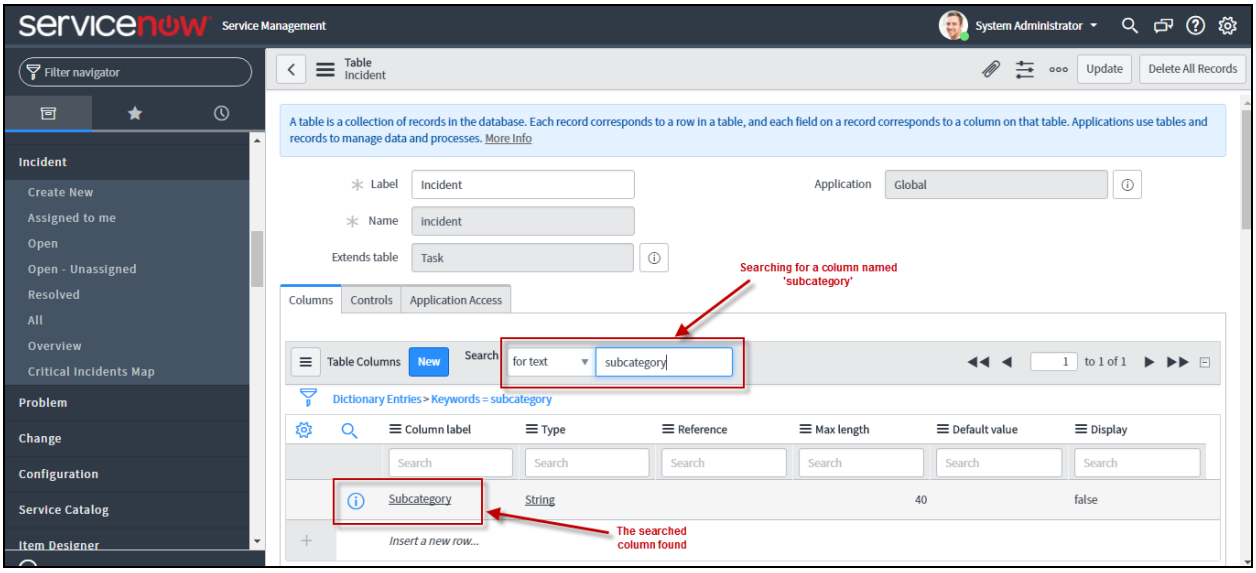

Figure 5.42: Clicking on the column 'contact'

- This will open Figure 5.43. The value of the **Column name** field will be displayed, as shown by Figure 5.43. As you can see, it is *subcategory*. This value should be used as the column name of the 'Subcategory' custom field in our example.

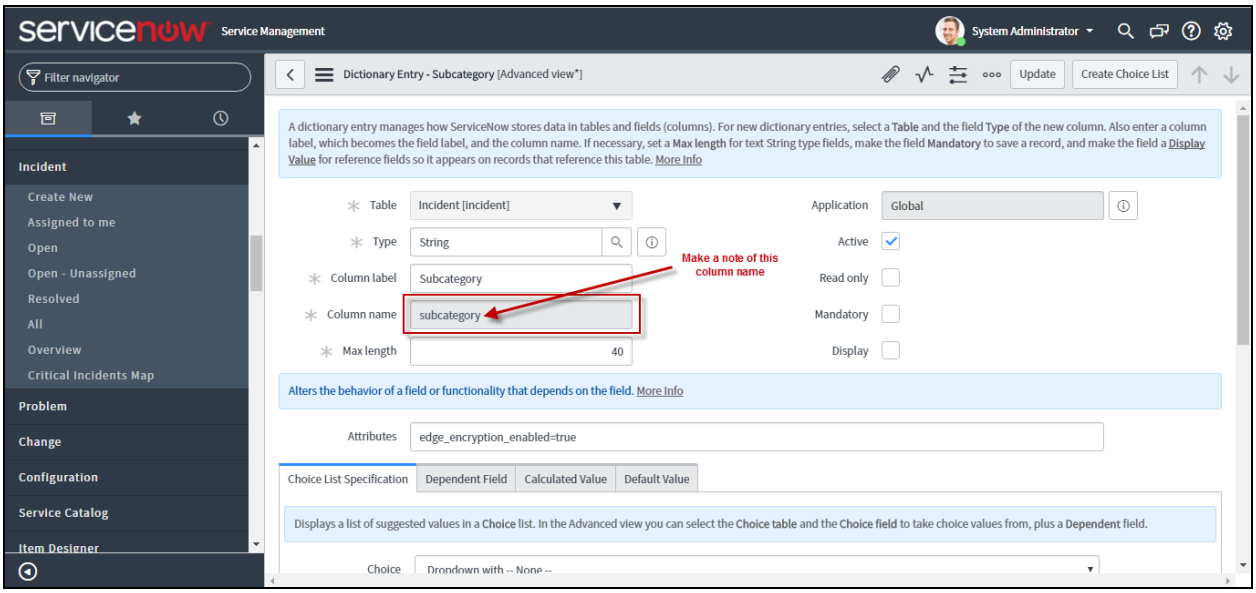

Figure 5.43: The column name of the 'Contact' field

2. Next, edit the **eg_services.ini** file (in the <EG_INSTALL_DIR>\manager\config directory). Look for the **SERVICE NOW** key in the **[TT_INTERGRATION_SYSTEM_FIELDS]** section of the file. A comma-separated list of parameters currently available in the **Trouble Ticket Integration** page for Service Now will be displayed by default against the **SERVICE NOW** key in this section. Add the new field, subcategory, to this list. **Note that the field will be displayed in the Trouble Ticket Integration page in the same position in which you insert the field here**. In the sample entry below, you can see that the new field has been inserted just before the **Critical due period** field.

```
SERVICE
NOW=url,port,user,password,isProxyEnabled,proxyHostName,proxyPort,isProxyAuthenticated
,proxyUserName,proxyPassword,caller_          id,assignment_          group,assigned_
to,category,subcategory,subcategory,criticalDueDatePeriod,majorDueDatePeriod,minorDueD
atePeriod,ticketTitleFormat,ttIntegratorClass
```

3. Next, look for the **[TT_INTEGRATION_FIELD_MAPPING]** section of the file. Under this section, add an entry of the following format:

```
<columnname>=<fieldtype>
```

```
<fieldtype> represents the type of values the new field will support. In the case of
our example, this entry will read as follows:
```

```
subcategory=Textfield
```

4. Then, go to the **[TT_INTEGRATION_TEXT_MAPPING]** section and add an entry of the following

format to it:

<columnname>=<displayname>

<displayname> represents the label using which the new field will be displayed in the Trouble Ticket Integration page.

In the case of our example, your specification can be:

subcategory=Subcategory

5. Finally, go to the [tt_integration_service now] section and add an entry of the following format to it:

   <columnname>=<defaultvalue>

   <defaultvalue> denotes the value that will be displayed by default against the new field in the Trouble Ticket Integration page. You can either provide a valid value or leave it as unconfigured. Let us leave it as unconfigured for the purpose of our example. The specification will therefore be:

   subcategory=$unconfigured

   In case you want to provide a valid value, then your specification will be:
   subcategory=Internal application

6. Finally, save the file.
7. If you now visit the Trouble Ticket Integration page for Service Now, you will find that the **Subcategory** field has appeared just above the **Critical due period** field (see Figure 5.44).

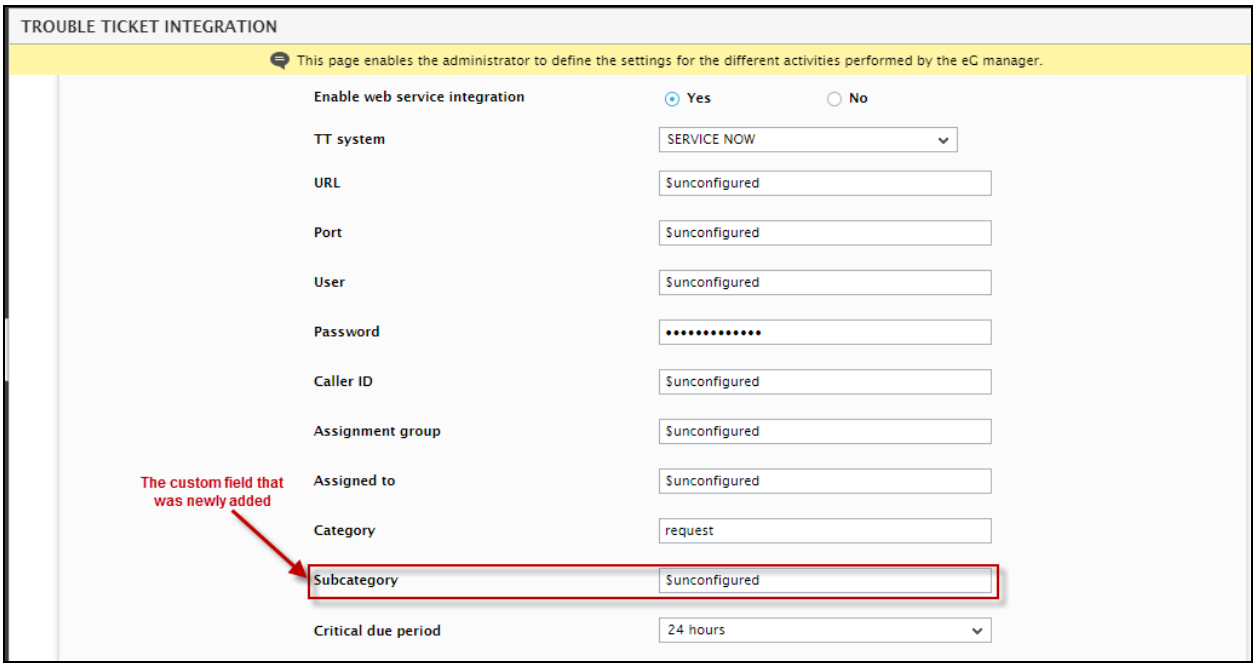

Figure 5.44: The custom field appearing in the Trouble Ticket Integration page of Service Now in the eG admin interface

5.20.2 Adding Custom Fields for JIRA Integration

Only custom fields created in JIRA can be added as custom parameters to the Trouble Ticket Integration page of the eG admin interface. This means that fields that pre-exist in JIRA cannot be included in the Trouble Ticket Integration page for JIRA in the eG admin interface.

Let us say that a custom field named 'eg manager url' has been added to JIRA. This field will contain a value only for eG alerts – not for alerts from any other source. With this information displayed alongside alerts, help desk staff will not only know where the alerts are coming from, but will also know where to go for more details about the problem.

To make sure that the 'eg manager url' field in JIRA is populated with a valid eG manager URL, the eG alerts sent into JIRA should also carry the URL of the eG manager that generated the alerts in the first place. For this to happen, let us add a custom field named 'eG manager URL' to the Trouble Ticket Integration page for JIRA in the eG admin interface, and then configure it with a static and valid eG manager URL.

The steps to be followed in this regard are discussed hereunder:

1. Find the ID that JIRA internally assigns to its custom field, 'eg manager url'. For this, follow the steps below:

   - Login to the JIRA console. Figure 5.45 will then appear. Click on the icon on the toolbar at the right, top corner of the JIRA console to invoke a drop-down menu. From the menu, pick the Issues option.

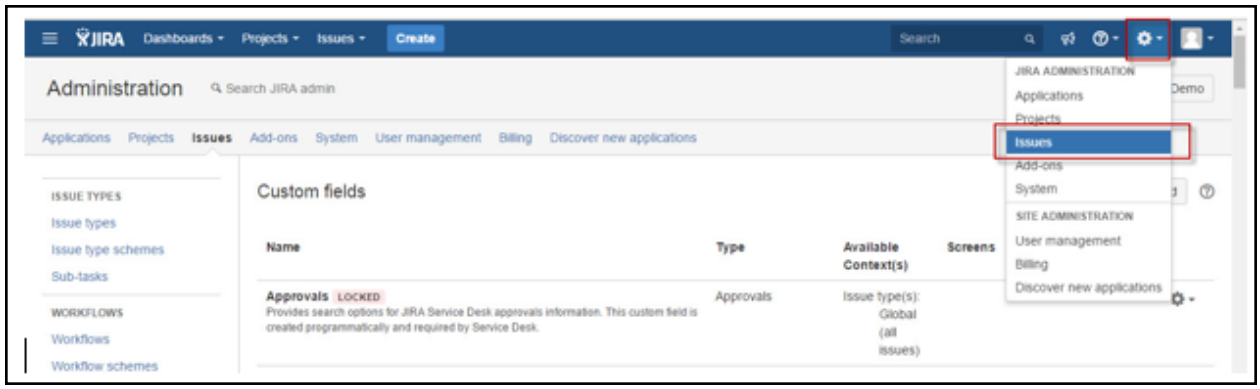

Figure 5.45: Selecting the Issues option

   - When Figure 5.46 appears, click on the Custom Fields option under fields.

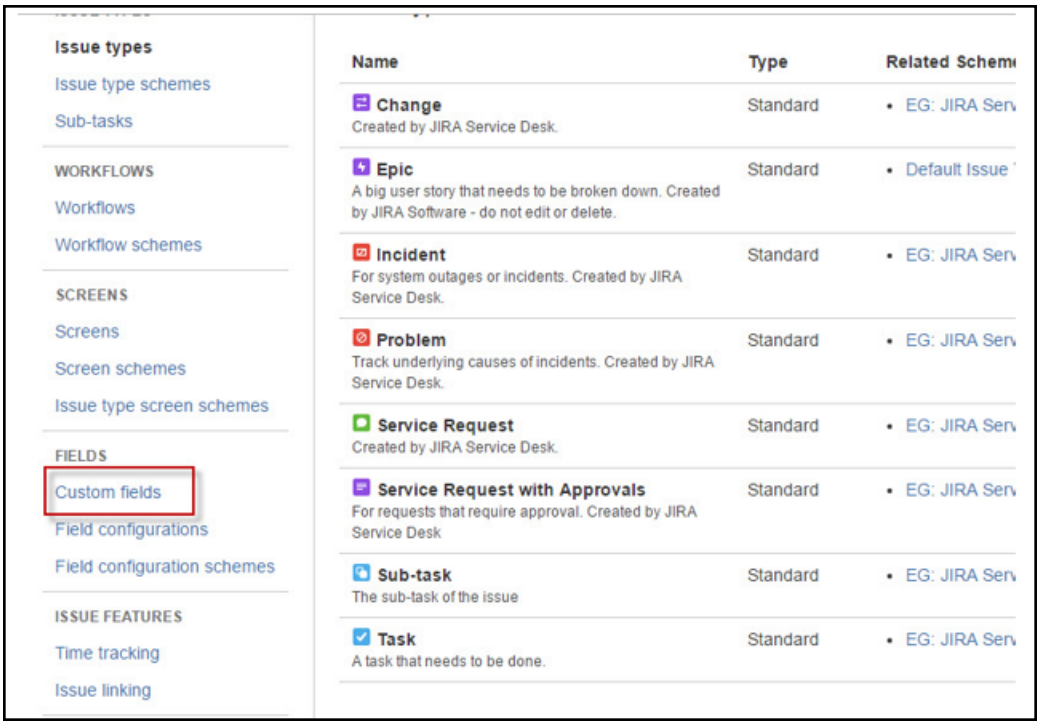

Figure 5.46: Clicking on the Custom Fields option

- Figure 5.47 will then appear listing all the custom fields that have been created in JIRA. Browse the list until you locate the 'eg manager url' field.

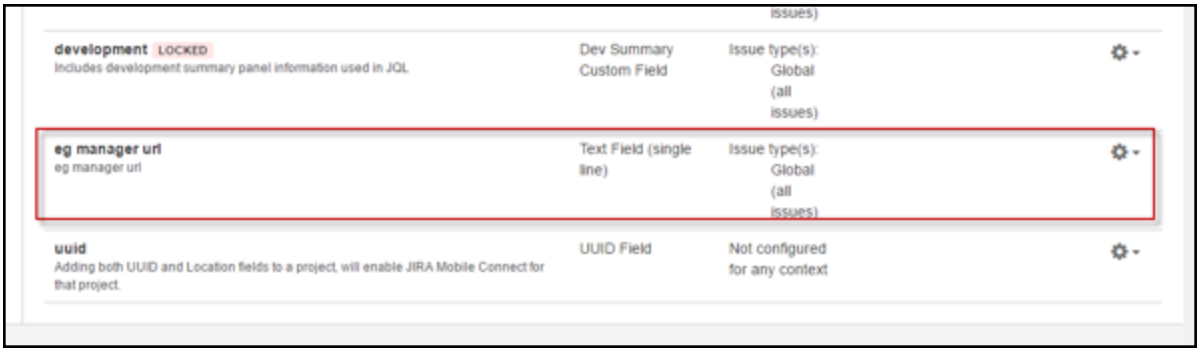

Figure 5.47: Locating the 'eG manager URL' field

- Next, click on the ⚙ icon corresponding to the 'eG manager url' custom field, and select the Edit option from the drop-down menu.

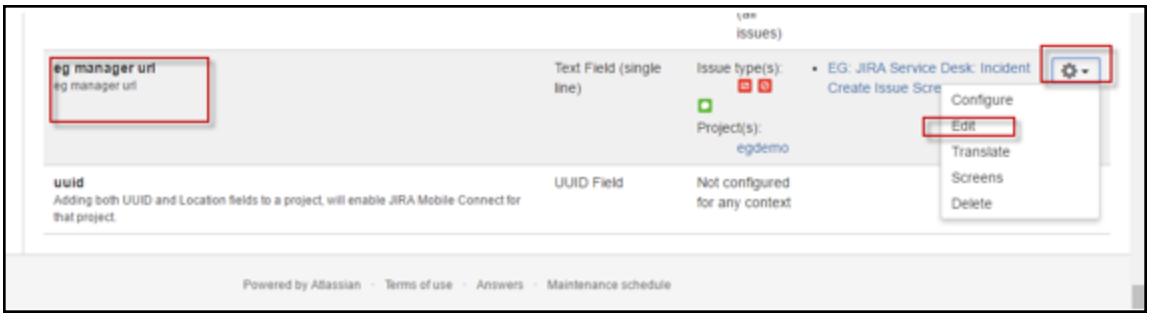

Figure 5.48: Selecting the Edit option from the drop-down menu

- When Figure 5.49 appears, shift your focus to the browser address bar. In the URL that appears in the address bar, look for the following entry: *EditCustomField!default.jspa?id=*

  The same has been indicated by Figure 5.49.

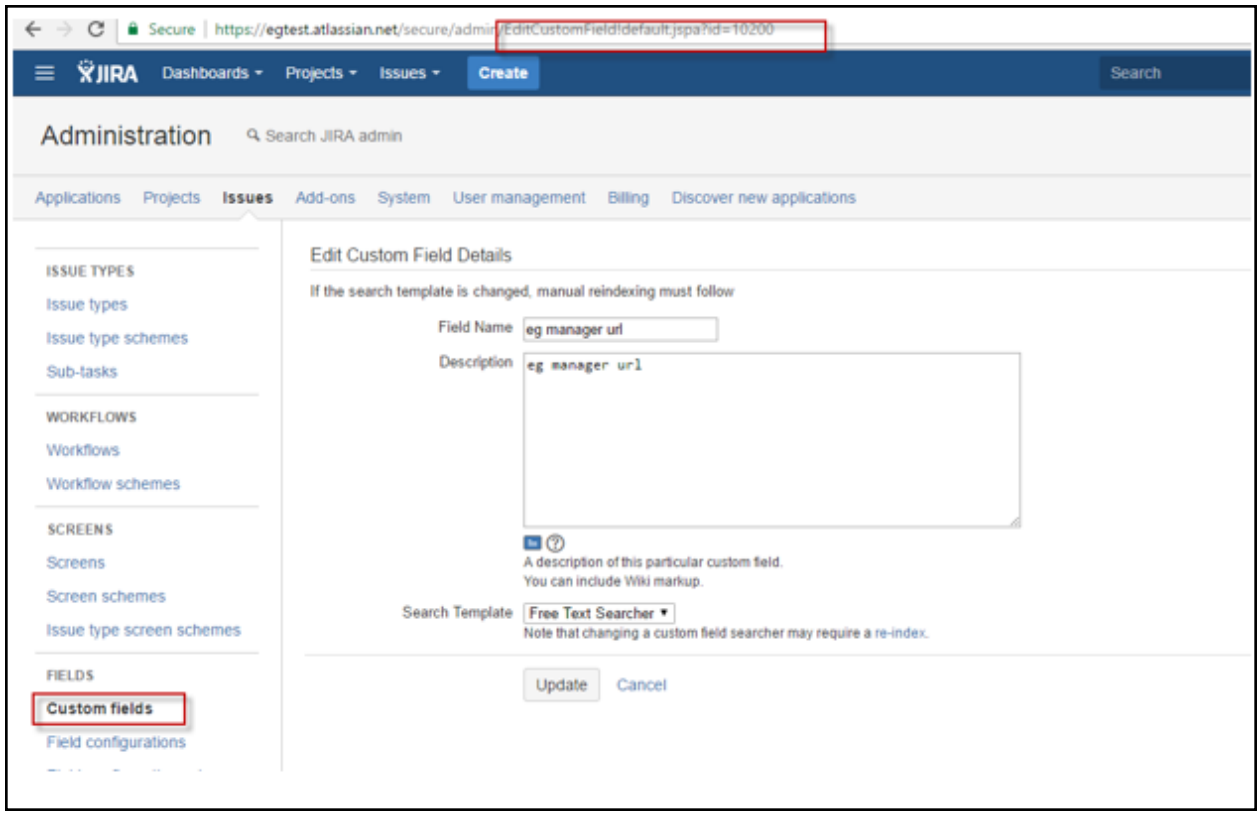

Figure 5.49: Determining the ID of the 'eG manager URL' custom field

The value that appears after the = symbol in the entry, is the ID of the custom field that is being edited – i.e., the ID of the 'eG manager url' custom field in our example. In Figure 5.49, this ID is 10200. Make a note of this ID.

2. Next, login to the eG manager host, and edit the **eg_services.ini** file (in the <EG_INSTALL_DIR>\manager\config directory). Look for the JIRA key in the **[TT_INTEGRATION_SYSTEM_FIELDS]** section of the file. A comma-separated list of parameters currently available in the Trouble Ticket Integration page for JIRA will be displayed by default against the JIRA key in this section. Insert the ID of the new field anywhere into the list of parameters that are pre-defined against the **JIRA** key. Since the field is a custom field, you need to add the prefix customfield_ to the ID 10200. This means that your custom field specification will be, *customfield_10200*. Note that the field will be displayed in the Trouble Ticket Integration page in the same position in which you insert the field here. In the sample entry below, you can see that the new field has been inserted just before the **ttIntegratorClass** field.

```
JIRA=url,user,password,projectKey,issueType,TitleFormat,DescriptionFormat,customfield_10200,ttIntegratorClass
```

3. Next, look for the **[TT_INTEGRATION_FIELD_MAPPING]** section of the file. Under this section,

add an entry of the following format:

<customfieldID>=<fieldtype>

<fieldtype> represents the type of values the new field will support. In the case of our example, this entry will read as follows:

customfield_10200=Textfield

4. Then, go to the **[TT_INTEGRATION_TEXT_MAPPING]** section and add an entry of the following format to it:

   <customfieldID>=<displayname>

   <displayname> represents the label using which the new field will be displayed in the Trouble Ticket Integration page.

   In the case of our example, your specification can be:

   customfield_10200=eG Manager URL

5. Finally, go to the **[TT_INTEGRATION_JIRA]** section and add an entry of the following format to it:

   <customfieldID>=<defaultvalue>

   <defaultvalue> denotes the value that will be displayed by default against the new field in the Trouble Ticket Integration page. You can either provide a valid value or leave it as unconfigured. Since we know the URL of the eG manager that is integrating with JIRA, let us provide a valid value here. For the purpose of our example, let this value be: http://192.168.9.244:7077

   The specification will therefore be:

   customfield_10200=http://192.168.9.244:7077

6. Finally, save the file.
7. If you now visit the **Trouble Ticket Integration** page for JIRA, you will find that the eG Manager URL field therein (see Figure 5.50).

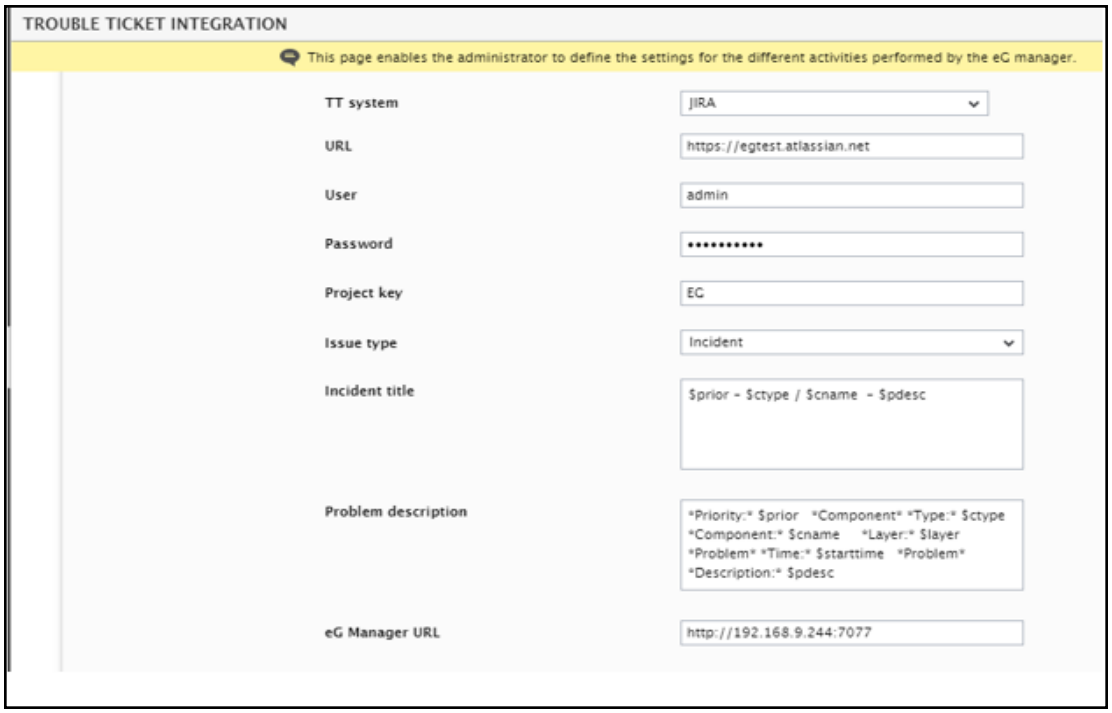

Figure 5.50: The custom field appearing in the Trouble Ticket Integration page of JIRA in the eG admin interface

5.20.3 Adding Custom Fields for Pager Duty, Hip Chat, and Slack

As mentioned earlier, custom fields can be added in the eG manager for only the following TT systems: Service Now, JIRA, Pager Duty, HipChat, and Slack. The procedure for configuring a Custom field in eG Enterprise for Service Now and JIRA are detailed in Section **5.20.1** and Section **5.20.2** (respectively). As for the other TT systems (Pager Duty, HipChat, and Slack), you can use the same procedure detailed for Service Now in Section **5.20.1**. The only differences will be:

- The procedure for finding the internal column name that corresponds to a custom field will vary from one TT system to another. Refer to the corresponding TT system documentation to know how to find the column name.
- A few section names and parameter names in the **eg_services.ini** file will also change according to the TT system. The table below should help you with that:

Parameter Name	Service Now	**SERVICE NOW** parameter in the **[TT_INTERGRATION_SYSTEM_FIELDS]** section
	Hip Chat	**HIP CHAT** parameter in the **[TT_INTERGRATION_SYSTEM_FIELDS]** section

	Slack	**SLACK** parameter in the parameter in the **[TT_ INTERGRATION_SYSTEM_FIELDS]** section
	Pager Duty	**PAGER DUTY** parameter in the **[TT_ INTERGRATION_SYSTEM_FIELDS]** section
Section Name	Service Now	**[TT_INTEGRATION_SERVICE NOW]** section
	Hip Chat	**[TT_INTEGRATION_HIP CHAT]** section
	Slack	**[TT_INTEGRATION_SLACK]** section
	Pager Duty	**[TT_INTEGRATION_PAGER DUTY]** section

Chapter 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.