

HP OMi Management Pack

For the Linux and Windows® operating systems

Software Version: 1.00

Development Guide

Document Release Date: January 2014

Software Release Date: January 2014



Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

© Copyright 2014 Hewlett-Packard Development Company, L.P.

Trademark Notices

Adobe® is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft group of companies.

UNIX® is a registered trademark of The Open Group.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to: <http://h20230.www2.hp.com/selfsolve/manuals>

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to: <http://h20229.www2.hp.com/passport-registration.html>

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support Online web site at: <http://www.hp.com/go/hpssoftwaresupport>

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

<http://h20229.www2.hp.com/passport-registration.html>

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

HP Software Solutions Now accesses the HPSW Solution and Integration Portal Web site. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this Web site is <http://h20230.www2.hp.com/sc/solutions/index.jsp>

Contents

Contents	3
Chapter 1: About this Guide	5
How this guide is organized	5
Who should read this guide?	5
Related Documentation	6
Abbreviations used in this guide	6
Chapter 2: Introduction to OMi MP	8
Chapter 3: Development Flow	13
Chapter 4: Designing an OMi MP	14
Golden Rules for Creating OMi MP	14
Application Domain Study	15
Studying the Application Domain	15
Studying the Possible Deployments of the Application	17
Studying Monitoring Metrics and Related Interfaces	18
Modeling Application Domain	19
Identifying CITs	19
Identifying relationships between CITs	20
Identifying Basic Application View	22
Identifying Health Components	23
Identifying Event Type Indicators (ETIs)	23
Identifying Health Indicators (HIs)	24
Identifying Monitoring Components	26
Identifying Aspects	26
Identifying Management Templates	28
Identifying Policy Templates	30
Chapter 5: Developing an OMi MP	33
Creating Modeling Components	33
Creating Health Components	40
Creating Monitoring Components	44

Creating an RTSM Package 61

Packaging OMi MP 63

Validating OMi MP 65

Appendix: Discovery 69

Glossary 79

We appreciate your feedback! 80

Chapter 1: About this Guide

This guide provides an overview of the development process of the HP Operations Manager i Management Pack (OMi MP), and provides a detailed information about creating an OMi MP.

How this guide is organized

This guide contains the following:

- [Introduction to OMi MP](#)

This chapter provides the prerequisites and references helpful in understanding OMi MP and Operations Manager i (OMi) terminologies used in this guide.

- [Development Flow](#)

This chapter provides a pictorial representation of creating an OMi MP and the information about the tasks that must be followed for developing a basic OMi MP.

- [Designing a Basic OMi MP](#)

This chapter provides information about designing and identifying the components for a basic OMi MP.

- [Developing a Basic OMi MP](#)

This chapter describes the following:

- Step-by-step instructions to create an OMi MP by using an example of PostgreSQL database
- Step-by-step instructions to package an OMi MP components into a single installable package

- [Appendix](#)

This chapter provides discovery information required for developing OMi MP.

Who should read this guide?

This guide is intended for the following users:

- A Subject Matter Expert (SME), who designs the monitoring solutions for enterprise applications using OMi
- An IT Operations monitoring developer

- A community content developer for OMi
- An HP Partner, who develops an OMi content

As one of these users, you will be familiar with Business Service Management (BSM) and the fundamental concepts of enterprise monitoring and management.

Related Documentation

For more information about related topics, see the following table:

Guide	Reference
BSM Release Notes BSM Installation Guide BSM Hardening Guide BSM Database Guide BSM Application Administration Guide BSM Platform Administration Guide BSM User Guide BSM Online Help BSM Modeling Guide BSM Extensibility Guide Operations Manager i Concepts Guide HP OMi Management Packs Installation Guide Monitoring Automation for HP Operations Manager i Installation Guide Monitoring Automation Release Notes HP Operations Agent and HP Operations Smart Plug-ins for Infrastructure Installation Guide	For more information, see the <i>BSM documentation</i> .
Infrastructure Content Pack User Guide Hadoop Content Pack User Guide Vertica Content Pack User Guide	For more information, see https://hpln.hp.com/

Abbreviations used in this guide

The following table provides information about the abbreviations used in this guide:

Abbreviations	Description
OMi MP	Operations Manager i Management Pack
MP	Management Pack
SME	Subject Matter Expert
BSM	Business Service Management
TBEC	Topology Based Event Correlation
MTTR	Mean Time To Resolve
CI	Configuration Item
CIT	Configuration Item Type
RTSM	Run-Time Service Model
ETI	Event Type Indicator
HI	Health Indicator
KPI	Key Performance Indicator
DFM	Data Flow Management
OEM	Original Equipment Manufacturer
JMX	Java Management eXtensions
WMX	Windows Management Instrumentation
uCMDB	Universal Content Manager Database
SLA	Service Level Agreement
OOTB	Out Of The Box
MA	Monitoring Automation
UI	User Interface
MS SCOM	Microsoft System Center Operations Manager
TQL	Topology Query Language
CMAs	Custom Message Attributes

Chapter 2: Introduction to OMi MP

OMi MP is the add-on content on top of OMi. It provides automatic and end-to-end monitoring solution of infrastructure and applications. OMi MP enables the users to proactively monitor, detect, troubleshoot, and remediate issues in the IT domain. It increases the productivity of the user by optimizing and automating various tasks, and reduces the mean time to resolve (MTTR) incidents.

It has the ability to discover an application domain and proactively monitor the domain for availability and performance issues. It consists of Management Templates, Aspects, Policy Templates, Performances Graphs, Troubleshooting Tools, Auto Remediation Flows, and Topology Based Event Correlation (TBEC) rules.

Components of an OMi MP

This section provides information about the components that can be included into an OMi MP.

The Components are as follows:

1. Run-Time Service Model (RTSM) Components

■ Configuration Item types (CITs) and relationship types

Integrates a CI into the BSM Operations Management monitoring solution for which a related Configuration Item Type (CIT) is not setup in BSM by default. For a PostgreSQL database, it is necessary to do the following:

- Create the new CIT
- Identify its key attribute values
- Establish the relationships, such as membership, dependency, or composition relationships

When modeling CITs and relationship types, you must ensure that you are able to discover or create corresponding Configuration Item (CI) instances and relationships automatically. For more information, see ["Discovery" on page 12](#).

For more information about Modeling, see the following location:

- BSM Online Help: Extensibility Guide > Operations Management > Content Development
- BSM Modeling Guide: Modeling > CI Type Manager

■ Views (Views and View mappings)

Views enable the user to visualize the context of an event. A typical View shows a subset of CIs in the RTSM and their relationships with other neighborhood CIs. BSM enables users to

create new Views in the RTSM. In addition, View mapping must be included to ensure that the Views are available for selection and use. For example, in the Health Top View pane, CIs impacted by the event selected in the event browser must be mapped to at least one of the existing Views. Multiple Views are listed by precedence, and the View with the highest precedence is shown as the default View.

For more information about Views, see the following location:

- BSM Online Help: Extensibility Guide > Operations Management > Content Development
- BSM Modeling Guide: Modeling > Modeling Studio > Create a Pattern View
- BSM Online Help: Application Administration > Operations Management > Operations Console > View Mappings

2. Health Components

■ Event type indicators (ETIs) and mapping rules

The various monitoring solutions send events regarding the status and availability of CIs to BSM. The event subsystem generalizes these events into a common language using ETIs. ETIs are categorizations of events according to the type of occurrence. For example, Central Processing Unit (CPU) load passing a threshold. Any occurrences on the monitored system of a given type causing an Operations Management event must be assigned to the same ETI.

Each CIT can have multiple ETIs, which define occurrences on a CI that are of interest, and may be monitored by BSM. ETI values can be used to represent the events that have an impact on the CITs specified in the TBEC rule. These values can be used to create correlation rules.

For more information about ETI and ETI mapping rules, see the following location in the *BSM Online Help*:

Extensibility Guide > Operations Management > Content Development

■ Health indicators (HIs)

It is important to transform the received events into data that provides information about the service health of the CIs. This involves analyzing incoming events for the various CIT and creating HIs. HI values are used to represent the events that have an impact on the CITs specified in the TBEC rule. These values are used to create correlation rules.

HIs provide fine-grained measurements for the CIs that represent your monitored business elements and processes. While some HIs display business metrics such as backlog and volume, the others display the various measurements of performance and availability. HIs are assigned to health-based Key Performance Indicators (KPIs).

For more information about Health indicators, see the following location in the *BSM Online Help*:

Extensibility Guide > Operations Management > Content Development

■ **Key performance indicators (KPIs)**

KPIs are high-level indicators of CI performance and availability, which apply calculation rules to the data provided by HIs to determine the CI status. KPIs are calculated using the status of HIs, KPIs, or a combination of these. For example, you can specify a rule that sets the severity of the key performance indicator (KPI) to the worst severity status of any assigned HI, or to the average severity status of all child KPIs.

The value that results from the calculation is used to set a severity level for the KPI based on the KPI definitions. KPI severity can be one of the following:

- Normal
- Warning
- Minor
- Major
- Critical

The resulting measurement for the KPI is translated into a color-coded status indicator displayed in Service Health, where the color represents a more desirable or less desirable condition for the KPI.

For more information about KPIs, see the following location in the *BSM Online Help*:

Extensibility Guide > Operations Management > Content Development

3. **Monitoring Components**

■ **Management Templates**

Management Templates provide a complete management solution for an application or a service. Management Templates are the containers for Aspects. Each Aspect provides the ability to monitor one characteristic of one or more types of CI. By grouping the Aspects together, you can create a management solution for several CIs that are related to each other.

For more information about Management Templates, see the following location in the *BSM Online Help*:

Application Administration > Operations Management > Monitoring > Management Templates and Aspects

■ Aspects

Aspects are the containers for Policy Templates, instrumentation, and parameters. Each Aspect provides the ability to monitor one characteristic of a CI. Aspects can include Policy Templates that can provide both agent-based and agent-less monitoring of CIs.

For more information about Aspects, see the following location in the *BSM Online Help*:

Application Administration > Operations Management > Monitoring > Management Templates and Aspects

■ Policy Templates and Parameters

The following types of Policy Templates are available:

- ArcSight Logger
- ConfigFile
- Flexible Management
- Logfile Entry
- Measurement Threshold
- Node Info
- Open Message Interface
- Scheduled Task
- Service Auto-Discovery
- Service or Process Monitoring
- SiteScope
- SNMP Interceptor
- Windows Event Log
- Windows Management Interface
- XML File

Policy Templates can be parameterized. Parameters enable you to create Policy Templates that can be customized by other users. Each parameter corresponds to a variable in a Policy Template. A parameter provides the users of a Policy Template, the opportunity to specify the value of a variable, without the need to modify the Policy Templates.

For more information about Policy Templates and Parameters, see the following location in the *BSM Online Help*:

Application Administration > Operations Management > Monitoring > Policy Templates

■ **Discovery**

To benefit from the advanced health-based monitoring features for a new application, offered by a BSM monitoring solution running Operations Management, it is necessary to populate the RTSM with CIs. The Operations Management events must be mapped to the correct CIs in the RTSM.

There are multiple methods to create CIs and CI relationships:

- Create the CIs and CI relationships using discovery features of the RTSM and HP Data Flow Management (DFM)
- Create the CIs and CI relationships using discovery Policy Templates
- Create the CIs manually in the RTSM if you need to create a limited number of CIs, and that these CIs are stable in nature, and are not expected to change

For more information about Discovery, see the following location in the *BSM Online Help*:

Extensibility Guide > Operations Management > Content Development

■ **Instrumentations**

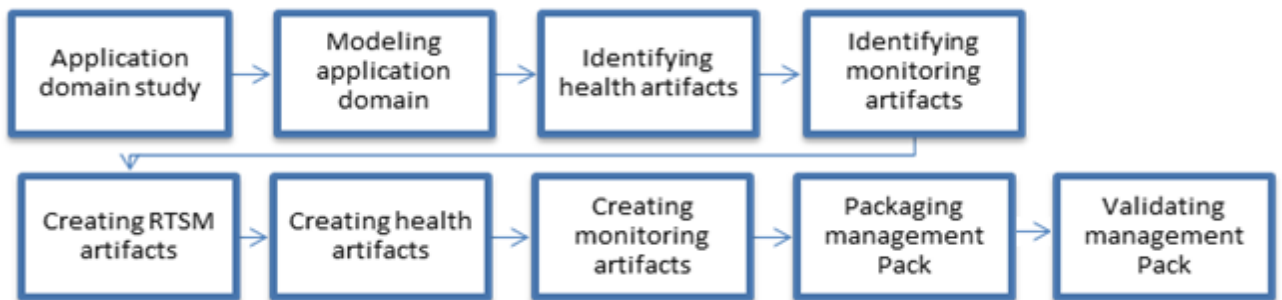
Instrumentation includes scripts or binaries that are needed by the HP Operations agent to discover and monitor the resources referred to in the Policy Template. The instrumentation collects the monitoring data and assesses the need of an alert to be raised.

Chapter 3: Development Flow

This chapter provides a pictorial representation of designing and developing an OMi MP. You can also see the following sections for more information:

- [Designing an OMi MP](#)
- [Developing an OMi MP](#)

The following high level steps are recommended while creating an OMi MP:



Chapter 4: Designing an OMi MP

This chapter provides information about designing and identifying the components for an OMi MP.

Golden Rules for Creating OMi MP

This section provides information about the rules for creating an OMi MP that can address the requirements of the customer.

- **Generate Actionable Events**

OMi MPs are created to enable users to solve problems effectively. Events that are generated by MP need to be highly actionable. If nothing needs to be fixed then do not raise an alert.

- **Enable first-level operators**

Enterprise customers want to enable the first-level operators by providing tools and relevant information so that they can work on the initial steps before assigning the ticket to SMEs. To enable the first-level operators, the following points must be considered:

- **Instruction text:** Instruction text is of primary importance as the first-level operators look for more information about the problem. Make sure that the instruction text is populated for each alert.

Instruction text consists of one or more of the following information:

- What could be the issue?
 - What is the next step?
 - Where to look for more information about the situation?
 - What should the user perform to know more about the situation?
- **Automatic action:** In the context of an event, you must consider automating actions. For example, if an important service is down, and can be restarted automatically, restart the service; send an alert only if the restart fails.
 - **Operator initiated action:** In the context of an event, help the user to get more information about the situation. For example, collecting more information about the processes from the machine where an event occurs will be helpful for the operator to include it as a part of the trouble ticket that is created for the event. This enables the SME to quickly get into the issue. Similarly you can consider embedding URLs or links to Performance Graphs that provides information on the availability and performance of neighborhood CIs.

- **Use readable message-texts**

The message texts must be meaningful, clear, and concise to ensure readability for the person who is on-call and reads the messages. The message must be readable on consoles and handhelds. Message format similar to the following can be used:

- **Event Template:** <Message text> <Current value> <Actual threshold value> for <Application Name><App Instance Name> <Object name><Object name value>
- **Sample message:** Elapsed time per execution (20 sec) has exceed threshold (10 sec) for ORACLE:ORCL01: SQL Query ID: 1234QWR”

- **Set the right severity**

Assigning the appropriate severity to the event is as important as generating actionable events. Severity of the events warrants the required attention of the user and can change the action of the user completely. Before setting a severity of an event, you must consider the following guidelines:

- Critical = Stop your current activity and fix the issue
- Major = Must be looked at in the next 6 to 12 hours; if not, may cause an outage
- Minor = Must be looked at in the next 12 to 24 hours
- Warning = Must be looked into within a week
- Normal = Acknowledge the issue

- **Avoid sending duplicate alerts**

You must ensure that the duplicate alerts are avoided as they are unnecessary in many cases. When a problem has not been resolved, you must ensure that the duplicate alerts for the same event are not sent to the operator console as each event generated incurs cost. The enterprise customers provide pager or equivalent devices to on-call support engineers. All the events that are generated, depending upon a priority, a ticket is raised and are sent to the device of the support engineer. Notifications in OMi are not triggered by duplicate events although OMi can detect duplicate alerts using certain techniques on the server side. As a best practice, it is recommended not to send duplicate alerts.

Application Domain Study

This is an important step where the application that you want to monitor is understood from the monitoring perspective. The following section guides you to study the application and get the required information that will help in creating the OMi MP for the application.

Studying the Application Domain

Before designing an OMi MP, you must understand the application components, the services

supported by the applications, and their relations. To design an OMi MP, follow these guidelines:

- What are the different application components of the application?
- What are the application components that you want to monitor?
- What are the application components that can be configured independently?
- What are the services that must be always running for the availability of the application?
- What are the services that are running when an optional application components are installed or enabled?
- What is the relation between services and different components of the applications?

For example, in a PostgreSQL Server environment, you can obtain the following information:

- PostgreSQL consists a client and server architecture
- PostgreSQL instance consists a set of Processes and Memory. PostgreSQL spawns process per-user. When a client requests for a connection to the database, the Postmaster gets the request; after performing authentication and authorization it spawns a new backend server process named **postgres**.
- PostgreSQL consists the following important processes:
 - The postmaster, supervisory daemon process
 - Back-end process per user postgres
 - Utility processes
 - Mandatory processes
 - BGWriter/Writer
 - WAL Writer
 - Optional processes
 - Stats-collector
 - Autovacuum launcher
 - Archiver
 - Syslogger
 - WAL Sender
 - WAL Receiver

Studying the Possible Deployments of the Application

To understand the environments that need to be supported by an OMi MP, you must analyze the recommended deployments by the Original Equipment Manufacturer (OEM) and understand the relation between application components and other domain components. This analysis will help you understand what environments need to be supported by an OMi MP and also enables you to create the required testing environments. Also, this enables you to understand how the health of the application is impacted by other applications or domains. For example, a PostgreSQL application is impacted when there is a problem in an underlying infrastructure domain.

To study the deployments of an application, follow these guidelines:

- What are the possible deployments recommended by OEM?
- When it is recommended to use each of the deployments?
- What are the supported high availability mechanisms?
- What are the supported cluster technologies?
- What are the hardware and software requirements recommended for each of the deployments?
- What are the other domain components that can affect the availability or performance of this application?

For example, if you consider a PostgreSQL deployed in a standalone and High Availability (HA) cluster mode, you can understand the environment and derive the following:

- Standalone: Single server hosting all processes and memory
- HA Cluster
 - Supports active or passive clustering where as Active or Active clustering is not supported
 - Maximum 16 nodes per cluster supported
 - Need to have a redundant system
 - Data redundancy
 - Network redundancy
 - Server and power redundancy
 - Enterprise customers deploy PostgreSQL in HA mode

Tip: It is important to understand the deployment scenarios from a customer perspective and support such scenarios.

Studying Monitoring Metrics and Related Interfaces

This section helps you to understand the various performance, availability, and configuration metrics of the application and interfaces that are created to collect them. To learn more about the monitoring metrics and related interfaces, follow these guidelines:

- What are the monitoring data metrics that are supported by the application?
- What are the key application metrics that are recommended to be monitored?
- What are the interfaces that are created to consume the data and metrics?
- What is the impact of potential downtime of the application on the business?
- What are the key log files? What is getting logged into it? What is the semantics of the log files?
- What are the credential requirements for accessing performance and availability metrics using these interfaces?
- Who are the end users of the application and the monitoring solution?
- What are the main issues faced by the end users of the monitoring solution?
- What are the metrics that need to be monitored for proactive problem detection before a particular problem occurs?
- What are the metrics that help to debug a given problem?

For example, in the PostgreSQL environment, PostgreSQL contains a statistics collector which is a subsystem that supports the collection and reporting of information about server activity. There are a set of tables and views where the database activity related data are captured. For more information, see *PostgreSQL documentation*.

Tip:

- In general, enterprise applications create the following metrics using standard interfaces:
 - Java Management Extensions (JMX)
 - Windows Management Instrumentation (WMI)
 - PowerShell
 - Performance counters
 - DB tables
 - Log files

- Users prefer to use lower privileged account for querying performance and availability, and it is required to understand the least privilege to get the required metrics.
- OEMs recommend a set of performance metrics to be monitored for a given application. It is important to monitor such metrics from OMi MPs.

Modeling Application Domain

This section provides information about identifying CITs and the relation between CITs.

Identifying CITs

This section helps you to identify CITs and its attributes for the application that you want to monitor.

The following section provides information about identifying CITs:

- Identify components in the application domain that need to be managed to deliver an IT service.
- There are several CITs that are available in the Universal Content Manager Database (uCMDB). You can derive a new CIT from appropriate base CIT. For example, PostgreSQL CIT needs to be derived from Database CIT.
- Each CI instance of the CIT should be distinguishable. Hence, items such as an instance of a database can be a CI, hence, a CIT can be created to define a database application. For example, we can define a CIT for PostgreSQL database. But, for queries and jobs that are intermittent, do not qualify to be a CI.
- Identifying key attributes is important for a CIT. After you decide to create a CIT, identify the key attribute(s) that uniquely identify a CI. It can be a single attribute or a combination of attributes that can be a key attribute.
- Define other important attributes of CIT. These attributes provide more information about the CI and help in the deployment of correct Aspects later. When you derive a CI from another CI, the derived CIs get the attributes from the parent CI. For example, PostgreSQL CIT gets attributes from Database CIT.
- Do not create additional CIs for any other reasons such as to create a link between CIs. This requirement must be handled by modeling relation between CITs.

Based on the guidelines, the following table provides information about the CITs that are identified for PostgreSQL:

CIT Name	Description
PostgreSQL Database	This CIT represents the PostgreSQL database
PostgreSQL Server	This CIT represents PostgreSQL server

Tip:

- What are the business objectives or SLAs that need to be met? What are the CIs that need to be monitored to ensure adherence to business objectives and SLAs? Model CITs to represent those CIs.
- Model what matters. You need not model all possible CIs. Model those CIs that need to be monitored and whose health really matters from availability and performance of the whole application ecosystem.
- Model to the right level. Model need not be too shallow or too deep. If you model too shallow, you might not be able to show which CI is exactly affected. If you model too deep, it will result with lots of CIs in uCMDB that might not add any value.
- Customers do not want to have more CIs that they do not configure in real time. Hence, understanding a real customer environment helps to understand the CIs that are getting configured.
- Model the CIs that can be discovered automatically.

For guidelines and best practices about modeling, see the following location in the *BSM Modeling Guide*:

Modeling > CI Type Manager

Identifying relationships between CITs

A model is not complete without appropriate relations between CITs. Relationships between CIs help the user to view a detailed information of the domain end-to-end. They also impact the way the health status is propagated. Hence, it is critical to model relationships between CITs correctly. This section helps you to identify relations between the CITs that have been identified.

- Now that the CITs are available, following questions help you to identify relationship between them:
 - How are the identified CITs related to each other?
 - How do these CIs interact?
 - Can these CIs exist independently of each other?
 - Is a CI part of another?
 - Is a CI member of another?
 - Is a CI dependent of another?
 - Can the performance or availability of a CI affect the other?

- Typically, relationships between CITs of the same domain can be depicted using the following relationships; but are not limited to these relationships:

- Composition
- Containment
- Membership
- Dependency

For example, the **PostgreSQL Server** consists of **PostgreSQL Database**, which cannot exist without **PostgreSQL Server**. Hence, choose the composition relation between these two CITs.

- When you want to establish relationships between the CIT of different domains, you must focus on the following questions:

- What are the other domains that are related to the domain that you are modeling?
- Need to look at both southward and northward of the CIT for relations. Typically health of the CI gets affected by health of a southward CI, and CI affects health of northward CI
- What are the other domain CIs that affect the health of the CIs that you are modeling? For example, the health of infrastructure CIs affects the health of PostgreSQL CIs
- Will the health of the CIs that are being modeled, affect the health of the CIs of other domain? For example, the health of PostgreSQL CI affects the health of Application CI that is dependent on the PostgreSQL database

- Typically relationships between CITs of different domains can be depicted using the following relationships; but are not limited to these relationships:

- Composition
- Membership
- Usage
- Dependency

- For example, let us consider an example of a Composition relation between the Node on which the PostgreSQL Server is running and the PostgreSQL Server and the PostgreSQL Database in the following table:

CIT 1	CIT 2	Relation
Node	PostgreSQL Server	Composition
PostgreSQL Server	PostgreSQL Database	Composition

Tip:

- Model and discover the relationships between CITs.
- Model the relationships that are important. Relationships are used for propagating health status and for topology based automatic correlation of events. Hence, model those relationships that effectively enable these features.
- Since most of the CIT inherit from base CITs, some of the relationships are automatically available.

For guidelines about the best practices for modeling, see the following location in the *BSM Modeling Guide*:

Modeling > CI Type Manager

Identifying Basic Application View

After modeling the CITs and relationships between them, you must create the required Views. One of the primary uses of the View is to show a subset of CIs from RTSM and the event browser for filtering events. This enables users to get detailed information of the situation and can understand the origination and root cause of the problem better. You must analyze the Views that are helpful for the operators and decide which View should be displayed when an event is selected from View mapping. Hence, it is important to have correct View(s) that displays CI and their relevant neighborhood CIs.

You must assign a View as a primary view for every important CIT. Primary view is rendered while the user selects an event. This help the user to look at the origin of the impact, as well as, from the current CI, where the impact propagates. You need to analyze, what are the CIs from both inter and intra domain that need to be shown from the context of any issue related to that CI. Hence, you need to add appropriate CITs from other domains to help the user to have this perspective.

Consider the following guidelines while identifying the basic application view:

- What are the health of the neighborhood CI that can impact the health of application CI?
- What are the other neighborhood CIs that affects the health of the application CI?
- What make sense to show from the CI perspective?
- What is the correct number of CIT that needs to be added, so that the user can quickly get into root cause of the issue?
- At run time, what will be the number of CIs that will be associated with the primary CI? Given this number, will it make sense to show that many number of CIs?

For example, by applying what has been discussed in this section, for PostgreSQL Server CIT, the primary view can contain the following CITs, PostgreSQL Server, PostgreSQL Database, and Node. Any performance and availability issues on node affects PostgreSQL Server and

PostgreSQL Database. For this example, let us name this View as **PostgreSQL Database Deployment View**. **PostgreSQL Server** is the primary CIT for this View.

Tip:

- You must ensure that optimal numbers of CITs are rendered when the View is run. Before adding CIT in a View, you must ensure the possible number of CIs present in the customer environment. This ensures the usability of the View.
- It is recommended to include a View that depicts the recommended application deployment environment.

For more information about View mappings, see the following location:

- BSM Modeling Guide: Modeling > Modeling Studio > Create a Pattern View
- BSM Online Help: Application Administration > Operations Management > Operations Console > View Mappings

Identifying Health Components

KPIs represent the actual status of the application and the IT infrastructure elements.

Identifying Event Type Indicators (ETIs)

ETIs are attributes of events (By default, the ETIs do not exist as instances). ETIs are used to categorize incoming events according to the type of occurrence in the managed IT environment. It needs to be defined per CIT. ETIs help to categorize events to abstract multiple event sources. ETIs are set based on a hint in event or using mapping rules. Topology Based Event Correlation correlates events based on their ETI and the CI relations stored in the RTSM.

The following section provides guidelines for identifying ETIs:

- Analyze various health aspects of CIs in your domain that can be affected by health of CIs from other domains. This analysis enables you to identify accurate ETIs that help in event correlation.
- Analyze the semantic of the event such as what does it mean? What type of event is this?

Abstract the meaning into an ETI name and state.

The following table provides information about some of the sample ETIs for PostgreSQL Database CIT:

ETI Name	Description	Possible Values
PostgreSQL Connection throttling	This indicator is set when it takes more time to get connected to a PostgreSQL and the total number of users connected and when the active connections are too high.	High Normal
PostgreSQL Transaction Rate	This indicator is set when the transaction rate of the database is high in terms of row updates, rollbacks, and deletes.	High Normal
PostgreSQL Disc Block Read Rate	Total disc blocks read per minute	High Normal
PostgreSQL error	An error is logged to a log file (Error does not affect the current health or the affect on health is unclear)	Occurred

Tip:

- Understanding the domain and important common occurring problems and understanding the root causes of those problems is the key to define useful ETIs and HIs.
- ETIs must be defined at correct granularity. The granularity should neither be fine-grained nor coarse-grained. If it is either way, it leads to too many ETIs that are not useful or too few, that do not facilitate meaningful event correlations. Hence, you must ensure not to have ETIs for every metric or have ETIs with tens of events mapped to it.
- The possible ETI values must convey a clear and concise message to the users.
- It is recommended to have limited number of ETI values. For example, two to three values.

Identifying Health Indicators (HIs)

HIs are ETIs that represent health. HIs are the link between events and health KPIs. HIs show detailed health of CIs. HIs help the user to identify different health aspects of a CI from availability and performance perspective and their current status. HIs are independent of the event life cycle. Operator can close corresponding events, but the real health of the CI is still shown. Operators can see the detailed health of a CI without having access to events. HIs are Domain-manager independent.

- A single unified CPU bottleneck HI can be set from different CPU-related events generated from OMi MPs, SiteScope or even from third-party domain manager such as Microsoft System Center Operations Manager (MS SCOM) using BSM connector
- To identify HIs, the similar guidelines are followed that are used to identify ETIs

- Health Indicator needs to clearly reflect the state of the health of the application CI from performance or availability. Hence, identify the health aspects of the application domain that the user wants to view in his day to day work
- Investigate key components of an application that you want to monitor, whose health is important for the overall application performance and availability
- HIs can display business metrics such as order backlog and volume, while others display various measurements of performance and availability
- To define an HI, verify the types of the event that are assigned to this HI. You can check the following:
 - What are the underlying metrics?
 - Whose status the event represents?
 - Does the status of those metrics reflect the health of the application?

If the preceding guidelines are not met, it does not qualify to be an HI.

The following table provides information about some of the sample HIs for PostgreSQL Database CIT:

HI Name	Description	KPI	Possible Values
PostgreSQL Connection Status	This health indicator reflects the status of connection to a PostgreSQL Database CI and any issue with the connectivity and the number of currently active connections to the database	Software Availability	Up Down
PostgreSQL Lock Status	Number of locks, waiting in the database	Software Performance	High Medium Low
PostgreSQL Cache Hit Ratio	The current ratio of buffer cache hits to total requests	Software Performance	Low Normal
PostgreSQL Query Performance	Indicates SQL statements with high elapsed time per execution	Software Performance	High Normal
PostgreSQL Transaction Health	Long running transactions and transaction failures	Software Performance	High Normal

The following table provides information about some of the sample HIs for PostgreSQL Server CIT:

HI Name	Description	KPI	Possible Values
PostgreSQL Server Connection Status	This health indicator reflects the status of connection to a PostgreSQL Server.	Software Availability	Up Down

Tip:

- HIs must be defined at correct granularity. Neither should be fine-grained nor coarse-grained. Else it might lead to too many HIs that are not useful or too few, that do not facilitate meaningful event correlations. In other words, should not have HIs for every metric or have HIs where tens of events are mapped.
- The possible HI values must convey a clear and concise message to the users.
- Each HIs need to have a normal value that can be mapped to green-state of the HI.
- It is recommended to have limited number of HI values. For example, two to three values.
- If you are defining new HIs, check whether the indicator qualifies to be an HI or an ETI.

Identifying Monitoring Components

This section provides information about identifying the following monitoring components:

- [Aspects](#)
- [Management Templates](#)
- [Policy Templates](#)

Identifying Aspects

Understanding the application domain is the key for coming up with relevant Aspects. The following guidelines will help you to identify Aspects:

- What are the predominant application deployments in the customer environment that you are targeting? What functional aspects are used, what components installed, how they are distributed?
- Identify the various functional aspects of the application domain. It is recommended to have the Aspects for important functional aspects.

- What are the components these functional aspects map to? Understanding this helps you to get correct granularity of Aspects.
- What are the functional aspects of the application? Among those what are mandatory and optional? In general, each of these functional aspects needs to be represented as Aspects. Aspects for optional functional aspects gets deployed only when appropriate components are present. This also helps when the user customizes the Management Template and do not want to have Aspects for optional components.
- What functional aspects (or components) can be installed together and the need to be installed separately? When there are mandatory components, which are always installed together and only few metrics needed to be monitored for them, then, you can consider having a single Aspect for those components.
- What are the functional aspects (or components) that the application SME or admin monitors? SME can choose to monitor a given Aspect, so the Aspects that you create needs to be in common parlance used by the application SMEs or admins.
- On a high availability mode, when a failover occurs, what components move across? In this scenario, deployment of Aspects automatically happens; hence, it is a good practice to have Aspects for those components to enable this movement.

The following table provides guidelines for Aspect naming:

Functional area getting monitored by an Aspect	Aspect naming template	Example
Availability/status of an application area	"<Application> [Area] Availability"	Oracle Availability, Vertica Availability
Performance of an application area	"<Application > <Area> Performance"	Oracle Query Performance, Oracle IO Performance
Database space	"<Application > <Area> Space"	Oracle Segment Space
Errors, failures, faults	"<Application > <Area> Faults"	Oracle Dataguard Faults
Monitors more than one above area	"< Application > <Area> <Health>"	Oracle Tablespace Health, Oracle Archiving Health

The following table provides information about PostgreSQL Server Aspects:

PostgreSQL Aspects	Description
PostgreSQL Database Availability	This Aspect monitors the status of database connection and various PostgreSQL processes

PostgreSQL Aspects	Description
PostgreSQL Memory Performance	This Aspect monitors the memory performance such as Buffer hits, Block reads, and Cache hit ratio.
PostgreSQL Query Performance	This Aspect monitors the inserts, updates, deletes, Index scans, Index fetches, Index read.
PostgreSQL Transaction Health	This Aspect monitors the commits and rollbacks.
PostgreSQL Cluster Health	This Aspects monitors the cluster services, cluster availability, heartbeat and replication

Tip:

- Aspects must be defined at correct granularity. Neither should be fine-grained nor coarse-grained. The fine-grained leads to too many Aspects that are useful and the coarse-grained leads to very few Aspects that will not enable the benefits of OMi.
- In general, the metrics-monitored part of single Aspect must affect a single HI or ETI.
- Each Aspect must be independent of the other. Deployment of one Aspect must not warrant for the deployment of another Aspect.
- By looking at the Aspect name and description, the SMEs must be able to understand what the Aspect monitors. To achieve this, follow the naming conventions provide adequate description for Aspects.

Identifying Management Templates

A Management Template includes several Aspects for monitoring the end-to-end functional components of an application domain. Management Templates are containers for Aspects. Each Aspect provides the ability to monitor an Aspect of a CI. By grouping Aspects together, you can create a management solution for several CIs that are related to each other. Essentially, the Aspects present in a Management Template must be related to the topology view of the application and the Management Template is assigned to its predominant CI type. The Aspects present in a Management Template is assigned to the CI types of the topology view. A topology path describing the way to find a CI type within the View is added. This design supports the service-centric model.

You can develop multiple Management Templates for monitoring an application domain. Each Management Template is an out-of-the-box (OOTB) management solution for an application domain, addressing a particular kind of monitoring requirement. Each Management Template comprises a set of Aspects.

The following section illustrates when to consider the multiple Management Templates for monitoring an application.

In an enterprise customer environment, there can be many Oracle instances where a portion of them might be hosting data for the business critical application while the remaining hosting the data for non-business critical applications. So, the user may choose to monitor them differently. Monitoring requirements can be different for these two categories of an Oracle instance from the following perspective:

- Metrics that the user wants to monitor for business critical and non-business critical applications is different. But, there is a common set of metrics that the user wants to monitor on both the categories.
- For the common set of metrics, the user wants to have different thresholds between business critical and non-business critical applications of an Oracle instances.

In this scenario, it is recommended to have two Management Templates catering to business critical and non-business critical application needs. One Management Template monitors only the essential metrics of an Oracle metrics with not-so-stringent thresholds, whereas the other Management Template is meant for monitoring business critical Oracle instance and monitors more metrics with stringent thresholds. This investigation enables you to decide the Aspects that need to be added as part of a given Management Template. Hence, to decide on how many Management Templates to have and what need to be part of those Management Templates, you need to analyze whether the user categorizes the instances of application CIs by criticality or otherwise. For example, in a service provider environment, it justifies monitoring each customer environment differently.

Create different Management Templates for providing agent based and agent less (SiteScope based) monitoring OOTB. It is also possible to have agent and agent-less monitoring capabilities in a single Management Template.

To create Management Templates, the following guidelines are recommended:

1. Before creating Management Templates, the following components should be identified:
 - Application model with CITs and relations
 - One or more Application topology views that depicts application deployment scenarios including CITs from other dependent domains like infrastructure
 - Aspects of the CITs that have been identified
2. Determine the number of Management Templates to be included
3. Choose Application topology view(s): Start with the application topology view that you have identified in previous steps. Choose the topology that you want to monitor using a Management Template. You may use single View for creating multiple Management Templates or different views for different Management Templates. Using a single or multiple Views depend on the application domain and the deployment. If the deployment scenarios are different and warrant for different root CITs, then it warrants for having different Views. In general, a single view that depicts the complex application deployment scenario will suffice for multiple Management Templates.

4. **Select Aspects:** For a given Management Template, select Aspects from the identified Aspects list. It is recommended to include Aspects for all the CITs that are part of the selected application topology view. Breadth and depth of the monitoring that you want to provide in a particular Management Template determines the number of Aspects in a Management Template.

For PostgreSQL, select **PostgreSQL Database Deployment View** for creating a Management Template. Create a Management Template with Aspects that monitors essential metrics and name it as **Essential Management Template for PostgreSQL**. This Management Template has Aspects mapping to PostgreSQL Database, PostgreSQL Cluster and System Infrastructure CIs. The Management Template is assigned to PostgreSQL Database CIT.

Tip:

- If the requirement is to monitor the same metrics across various instances CIT of an application environment and only the thresholds need to be different, then having a single Management Template will suffice. Different thresholds can be provided at the time of deployment.
- For monitoring an enterprise application, consider having a Management Template where end user can get optimal solution of using both agent and agent-less monitoring. The agent and agent-less technologies augment to provide better monitoring solution.
- For monitoring an application environment, it is recommended to have a Management Template that monitors only the golden metrics (Key metrics). Application OEM generally recommends a list of key metrics that need to be monitored to keep a tab on application availability and performance. This enables the user to quickly deploy Management Template in the environment with minimal customization.

Identifying Policy Templates

In the section [Studying monitoring metrics and related interfaces](#), you have identified the metrics and the how these metrics need to be accustomed (Interfaces or APIs or tables or Log file and so on). There are several Policy Templates available for various different requirements. For more information about the list of Policy Templates and their purpose, see the following location in the *BSM Online Help*:

Application Administration > Operations Management > Monitoring > Policy Templates

To collect metrics and raise events in threshold breaches, you must consider the following components:

Policy Template

A Policy Template is a set of configuration information for HP Operations Agent, HP SiteScope, or HP Arcsight Logger. Policy Templates define the details of specific configuration and monitoring tasks. Agent runs the instrumentation on a given interval and collects metrics, compares the value against thresholds, and raises events. In this section we will look at how to develop and deploy Policy Templates to computers that run the HP Operations Agent.

Instrumentation

These are binary or scripts that are invoked by the Policy Templates. These scripts can be external to policy when a metric needs to be collected. Logic to collect the performance and availability metrics from applications, to log the collected metrics and to perform value addition on top of collected metrics is present in instrumentation. Scripts can be external to Policy Templates or embedded in the policy. The instrumentation scripts are automatically deployed when Policy Templates are deployed to the CIs.

Parameters

Parameters enable end users to provide different values for the attributes that often get customized in the user environment. For example, if you have a Policy Template that monitors the level of CPU usage, you can have parameters for a minor event threshold, a major event threshold, and a critical event threshold. User can customize this by providing custom values for these parameters to specify the level of CPU usage that is a minor, major, or critical event on the computer that they want to monitor. The user need not have to modify the Policy Template and need not have detailed knowledge of how the Policy Template monitors the CPU. The user needs to know what monitoring functionality the Policy Template provides and the purpose of the parameters.

Parameters also enable you to create Policy Templates that use values you cannot specify in advance. For example, a Policy Template that monitors database performance might need a user name and password to connect to the database. Appropriate parameters provide a generic Policy Template, without the hard-coded user credentials. After a Policy Template is assigned and deployed, an application expert can customize the parameters and tune the monitoring solution.

An instance parameter enables you to create Policy Templates that monitor multiple instances of the same type of object (for example, multiple database instances or multiple hard disks). Each Policy Template can have only one instance parameter. After you add an instance parameter to a Policy Template, all other parameters become dependent on it. The user can specify separate values for the dependent parameters of each instance.

For example, consider the following:

- For PostgreSQL Database, you must collect the performance and availability metrics from the databases
- To obtain the metrics, you must run the SQL queries
- Logic to connect to PostgreSQL performance metric tables can be written as binary (`PostgresqlDBCollector.exe`), which can take metric that need to be collected as input. This binary is scheduled to be called every 15 minutes from a **Schedule task policy**.
- To capture the thresholds and check the metrics whether there are threshold breaches, **Measurement Threshold policy** must be used.
- There are multiple **Schedule Task policies** that run the executable `PostgresqlDBCollector.exe` with different metric names as input
- Logic present in the binary connects to PostgreSQL Database, collects the metrics from appropriate tables, sends the collected values to the corresponding Measurement Threshold policy which checks for the threshold breach using `opcmmon` command.

For more information about the syntax and the usage of `opcmon` command, see the following location in the *BSM Online Help*:

Application Administration > Operations Management > Monitoring > Policy Templates > Configuring Measurement Threshold Policies

The following table provides information about the possible Policy Templates and Aspects for PostgreSQL:

PostgreSQL Aspects	Policy Templates	Policy Template type
PostgreSQL Database Availability	PostgreSQLDB_Availability	Measurement Threshold
	PostgreSQLDB_Scheduler	Schedule Task
PostgreSQL Memory Performance	PostgreSQLDB_Buffer_Hits PostgreSQLDB_Block_Reads PostgreSQLDB_CacheHit_Ratio	Measurement Threshold
	PostgreSQLDB_Scheduler	Schedule Task
PostgreSQL Query Performance	PostgreSQLDB_Inserts PostgreSQLDB_Updates PostgreSQLDB_Deletes PostgreSQLDB_Index_Scans	Measurement Threshold
	PostgreSQLDB_Scheduler	Schedule Task
PostgreSQL Transaction Health	PostgreSQLDB_Commits PostgreSQLDB_Rolebacks	Measurement Threshold
	PostgreSQLDB_Scheduler	Schedule Task
PostgreSQL Cluster Health	PostgreSQL_Cluster_Service_Availability PostgreSQL_Cluster_Heart_Beat PostgreSQL_Cluster_Replication	Measurement Threshold
	PostgreSQLDB_Cluster_Scheduler	Schedule Task

Chapter 5: Developing an OMi MP

This chapter provides information about the steps involved in developing an OMi MP.

Creating Modeling Components

Before creating the components of an OMi MP, you must install BSM and OMi. For more information, see the *BSM Install Guide* and the *Monitoring Automation (MA) Install Guide*.


To create Modeling Components, follow these tasks:

Task 1: Creating CIT

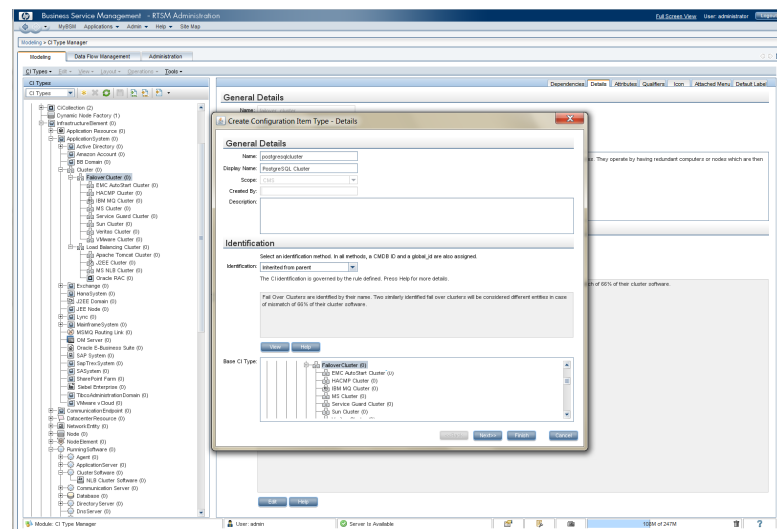
You must create new CITs in RTSM using CIT Manager. To create new CITs, follow these steps:

1. Click **Admin > RTSM Administration**.
2. Click **CI Type Manager**.
3. Inherit CIT from the appropriate root CIT.

Note: Since PostgreSQL is a database, CIT needs to be inherited from the base CIT database.

4. Right-click **Database**, and then click  **New**.

The Create Configuration Item Type Details window is displayed.



5. In the General Details section, provide the CIT values such as CIT Name, Display Name,

Scope, Created By, and Description.

6. In the Create Configuration Item Type-Details section, select the identification method from the Identification drop down list. By default, the identification method is selected as Inherited from parent.

Note: Identification rules are used to uniquely identify a CI.

7. Select Database as **Base CI Type**.

8. Click **Next**.

You can see the inherited attributes that are available for a specific CIT. You can edit or add new attributes. For PostgreSQL Server, continue with the default inherited attributes.

9. To modify an existing attribute, click **Edit**.

10. Click **Finish** to create CIT.

CIT **postgresql_server** is created with the display name **PostgreSQL Server**.

Task 2: Adding product name to product name enumeration

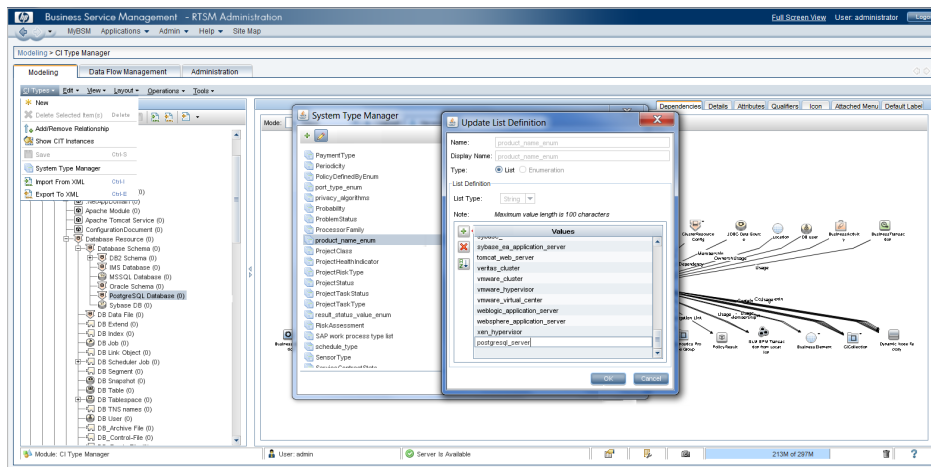
For the CITs that are derived from the Running Software type, the name of the CIT needs to be added to the product name enumeration. When a CI of that particular CIT is created in RTSM, the ProductName attribute of the CI needs to be set to this value. This is used to identify CIs of a particular product.

For PostgreSQL Server, add **postgresql_server** to the product name enumeration. To create product name enumeration, follow these steps:

1. Click **Modeling > CI Type Manager**, and then click **CI Types** in the menu and select **System Type Manager**.
2. In the System Type Manager window, select **product_name_enum**.
3. To modify, click **Edit** on the top left corner of the window.
4. To add a new item in the list, click **Add**.

A new empty item gets added to the bottom of the list.


5. Scroll down and type the product name that needs to be added.
6. Click **OK**.
7. Click **Apply**.

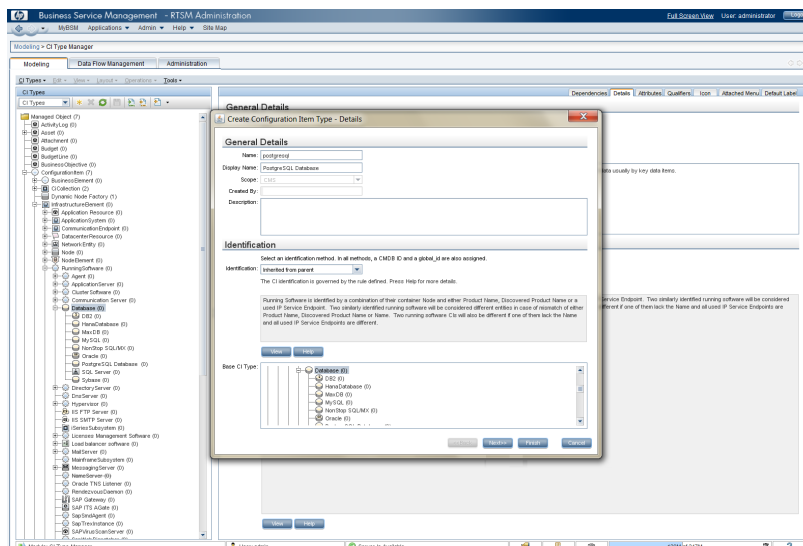


Task 3: Creating PostgreSQL Database CIT

This section provides information about creating PostgreSQL Database CIT. You must inherit the PostgreSQL from the Database Schema.

To create a PostgreSQL Database CIT, follow these steps:

1. Click **Admin > RTSM Administration**.
2. Select **CI Type Manager**.
3. Inherit CIT from appropriate root CIT.
4. Right-click **Database**, and then click  **New**.



5. In the General Details section, provide CIT values such as CIT Name, Display Name, Scope, Created By, and Description.
6. In the Create Configuration Item Type-Details section, select the identification method from

the Identification drop down list. By default, the identification method is selected as Inherited from parent.

Note: Identification rules are used to uniquely identify a CI.

7. Select Database as **Base CI Type**.

8. Click **Next**.

You can see the inherited attributes that are available for a specific CIT. You can edit or add new attributes. For PostgreSQL Server, go with the default inherited attributes.

9. To modify an existing attribute, click **Edit**.

10. Click **Finish** to create CIT.

CIT **postgresql_server** is created with display name **PostgreSQL Server**.

Task 4: Creating relationship between CITs

To create relationships between CITs, you must first create the relations that are discussed in the [Identifying relationships between CITs](#) section, and then create a Composition relation between the Node and the PostgreSQL Server.

To create a relation between two CITs, follow these steps:

1. Select the Node and the PostgreSQL Server from **CI Type Manager**.

Note: You can use **ctrl** key to select multiple CITs.

2. Right-click **Node** and **PostgreSQL Server**, and then select **Add/Remove Relationship** from the menu.

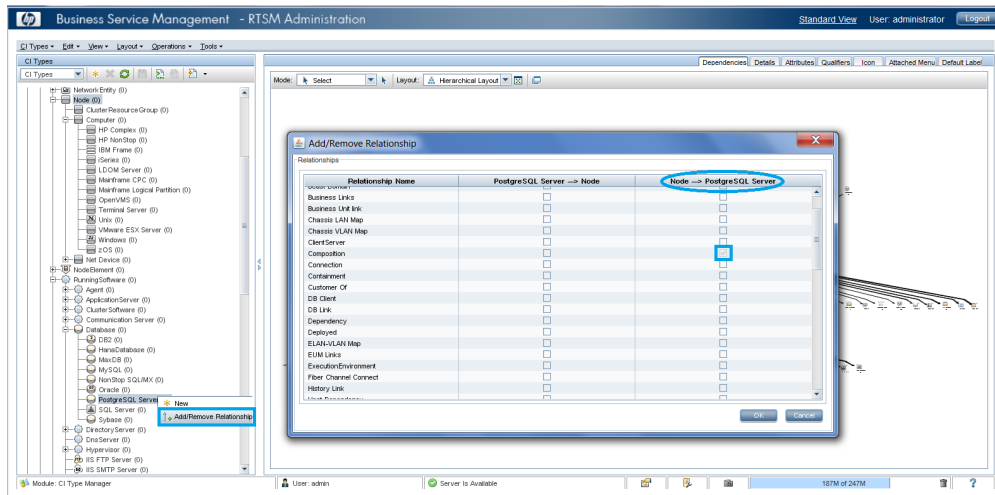
3. Select the check box for the appropriate relation that you want to have between CITs.

Note: Between the two CITs, it is possible to create relationship from either of them to other, ensure whether you are creating the relationship that you intended to create.

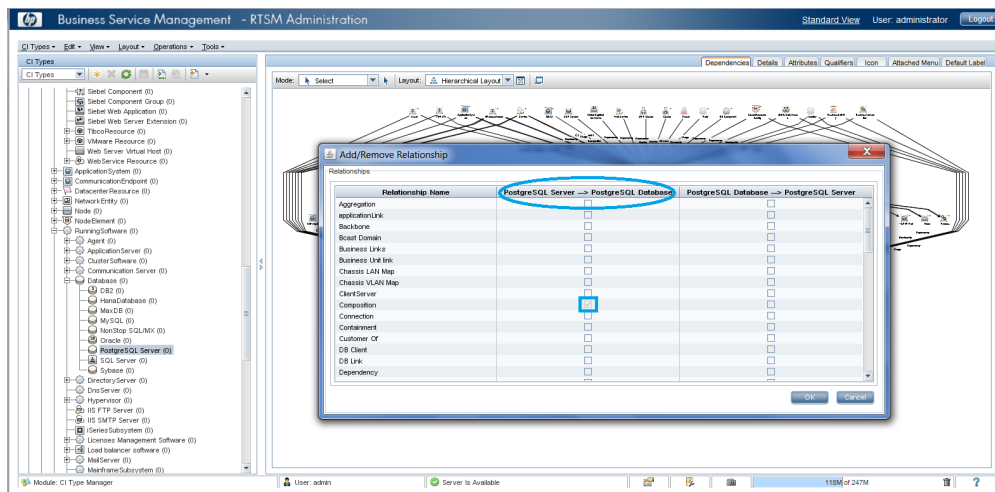
For PostgreSQL, you must create following relationships:

- Composition relation between Node and PostgreSQL Server
- Composition relation between PostgreSQL Server and PostgreSQL Database

Since the parents of these CITs have Composition relation defined, the node and the PostgreSQL Server have the composition already selected.



Similarly, the PostgreSQL Server and the PostgreSQL Database also have the composition already selected.




Task 5: Views and View mappings

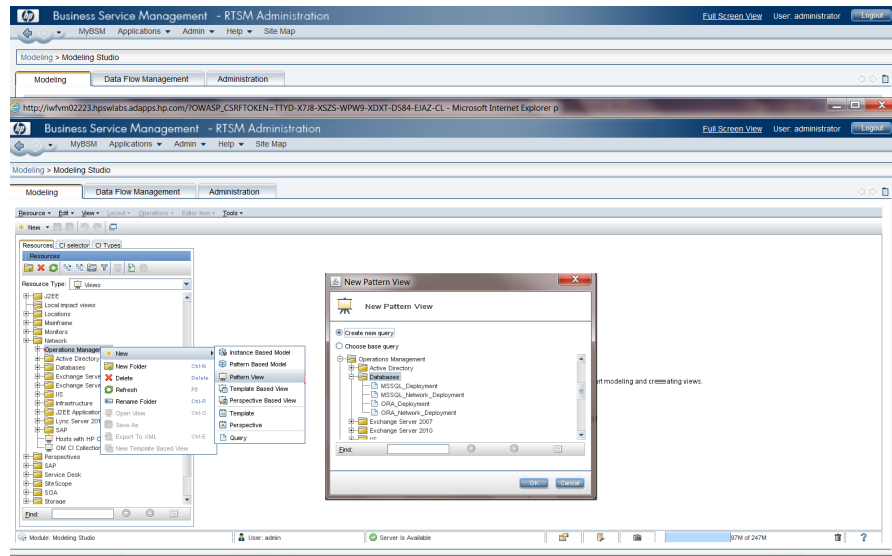
You must create the View for PostgreSQL. To create a View, follow these steps:

1. Navigate to RTSM Administration, and then click **Modeling Studio**.
2. Select **Views** in Resource Type drop-down box.

It is recommended to create the views under Operations Manager.

3. Right-click **Operations Manager** from the tree view, click  **New**, and then click **Pattern View**.
4. In the New Pattern View window, select **Database** under Operations Manager in the tree view.

5. Select **Create new query**.

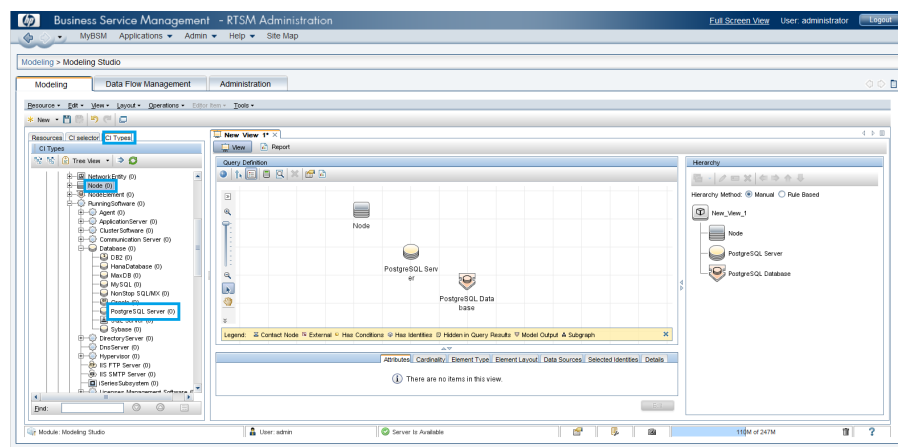


6. Click **OK**.

Task 6: Drag-and-Drop CITs

To drag-and-drop CITs, follow these steps:

1. Click the **CI Types** tab.
2. Drag-and-drop the following CITs into the center pane, to make them part of the View:
 - Node
 - PostgreSQL Server
 - PostgreSQL Database



The CITs are now available as a part of the View.

Task 7: Creating relations between CITs

To create relations between CITs in the View, follow these steps:

1. Select the CITs that you need to relate, right-click and select **Add relationship**.
2. In the Add relationship window, select the relation that needs to be part of the View.

You can name the relation.

The relations that are available between the CITs are displayed.

3. Select the **Node**, the **PostgreSQL Server**, and then select **Composition**.
4. Enter **Hosted On** in the Relationship Name field.
5. Click **OK**.

Setting cardinality: It is important to set correct cardinality to the CITs in the View. For each Node, there can be one or many PostgreSQL Servers. Similarly, for each PostgreSQL Server, there can be one or more PostgreSQL Databases. So, the appropriate cardinality is selected in the following screen shot. You can modify the cardinality by clicking **Edit** in the right bottom of the PostgreSQL Deployment View page.

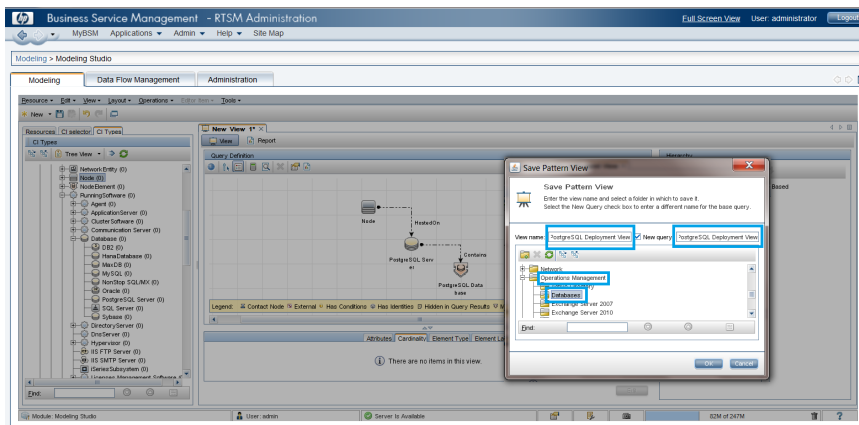
Similarly, the relationship and cardinality for other CITs needs to be created.

Task 8: Saving a View

After the View is complete, you can save it.

- Before saving a View, you can specify a name and you need to choose the location where a View needs to be saved.

For example, PostgreSQL View is saved as PostgreSQL Deployment View, under Operations Manager/ Databases. When a View is saved, the Topology Query Language (TQL) for the View is also saved with the same name. TQL can be viewed under Resource Type/Queries.




Task 9: Mapping a View to CIT:

After creating a View, it needs to be assigned to a CIT. This operation can be performed from View Mapping. To reach View Mappings, follow these steps:

1. Click **Admin > Operations Management**.
2. Select **View Mappings**.

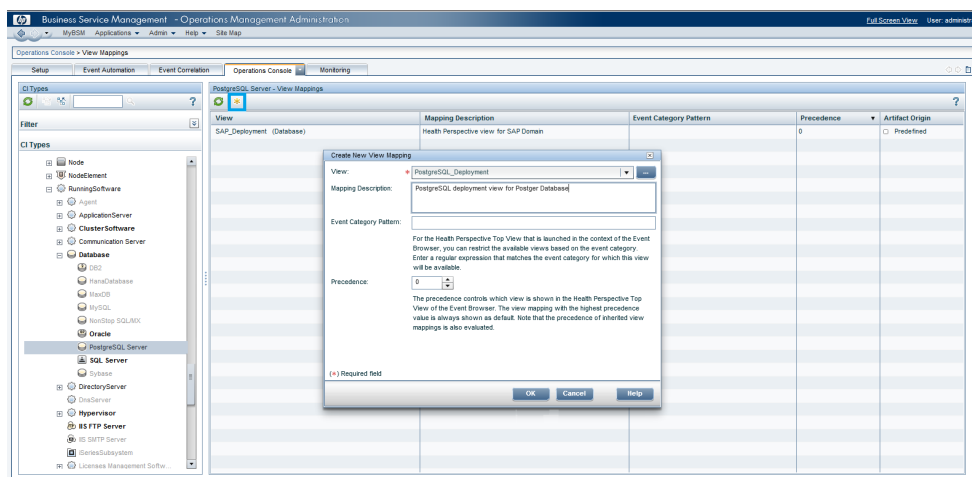
In the View Mappings window, follow these steps to assign a View to a CIT:

1. Select the CIT that need to be assigned a View.
2. Click  **New**.
3. In the Create New View Mapping window, select the View that needs to be assigned to the CIT.
4. Select the **Precedence** in the Precedence field.

Note: You can assign multiple precedence Views.

5. Click **OK** to save the View assignment.

For example, The CITs PostgreSQL Server and the PostgreSQL Database are assigned the PostgreSQL_Deployment View.



Creating Health Components

This section provides information about creating indicators.

Task 1: Creating ETIs and ETI mapping rules


This section provides information about creating Event type indicators.

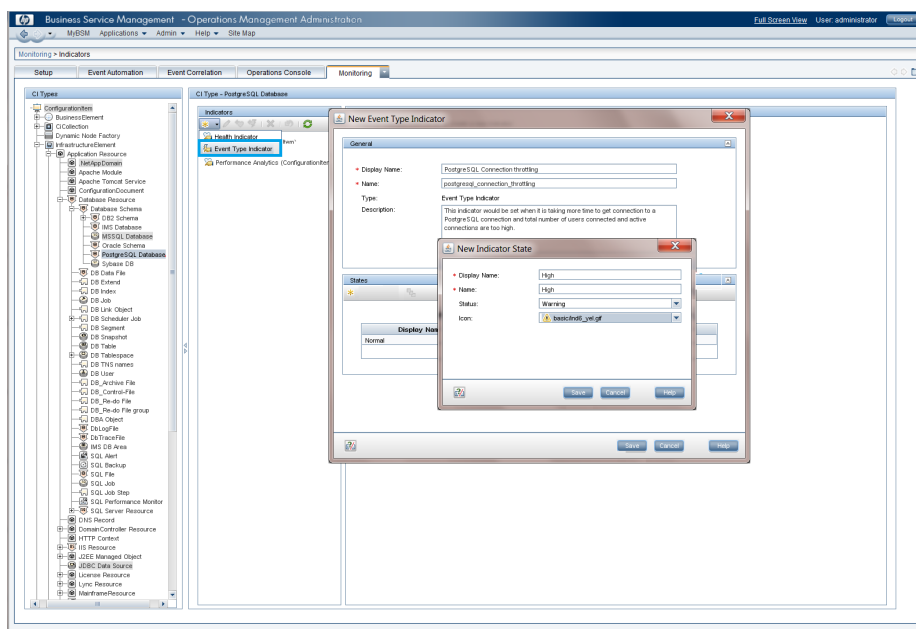
To access Indicators, follow these steps:

1. Navigate to Indicators creation page from **Admin\Operations Management** menu.
2. Click **Monitoring** tab, and then click **Indicators**.

In the resulting window, select the CIT for which you want to create ETI.

To create ETIs that are identified in the section [Identifying Event Type Indicators \(ETIs\)](#), follow these steps:

1. Click  **New**, and then select **Event Type Indicator**.
2. In the New Event Type Indicator window, provide the ETI details such as Display Name, Name, Type, and Description.
3. Click **New Indicator State**.
4. In the New Indicator State window, provide the state details such as Display Name, Name, Status, and Icon for the ETI.
5. Click **Save**. Repeat steps 3, 4, and 5 for all the states that you want to create for an ETI.
6. Click **Save** to save the ETI.




Task 2: Creating HIs

This section provides information about creating HIs.

To create HIs, follow these steps:

1. Navigate to Indicators creation page from **Admin\Operations Management** menu.
2. Click **Monitoring** tab, and then click **Indicators**.



In the resulting window, select the CIT for which you want to create HI.

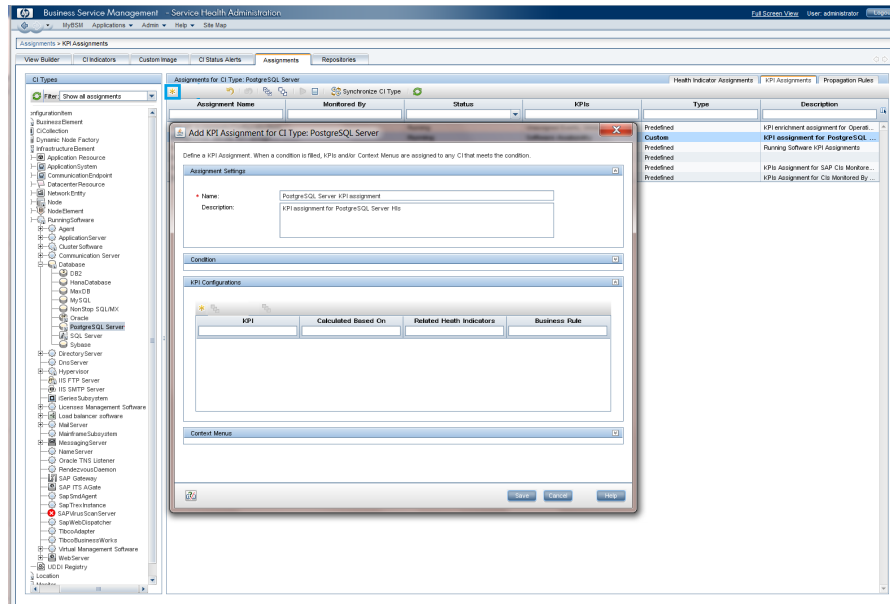
3. Click  **New**, and then select **Health Indicator**.
4. In the New Event Type Indicator window, provide HI details such as Display Name, Name, Type, and Description.
5. Click **New Indicator State**.
6. In the New Indicator State window, provide state details such as Display Name, Name, Status, and Icon for the HI.
7. Click **Save**. Repeat the steps 5, 6, and 7 for all the states that you want to create for a HI.
8. Click **Save** to save the HI.

Task 3: KPI assignment

To ensure that the KPIs are providing the correct status for a specific CIT, the HIs that are created should be assigned to the appropriate KPI. HIs meant for showing availability and performance health need to be assigned to **Software Availability** and **Software Performance** KPIs respectively.



To create KPI Assignments, follow these steps:

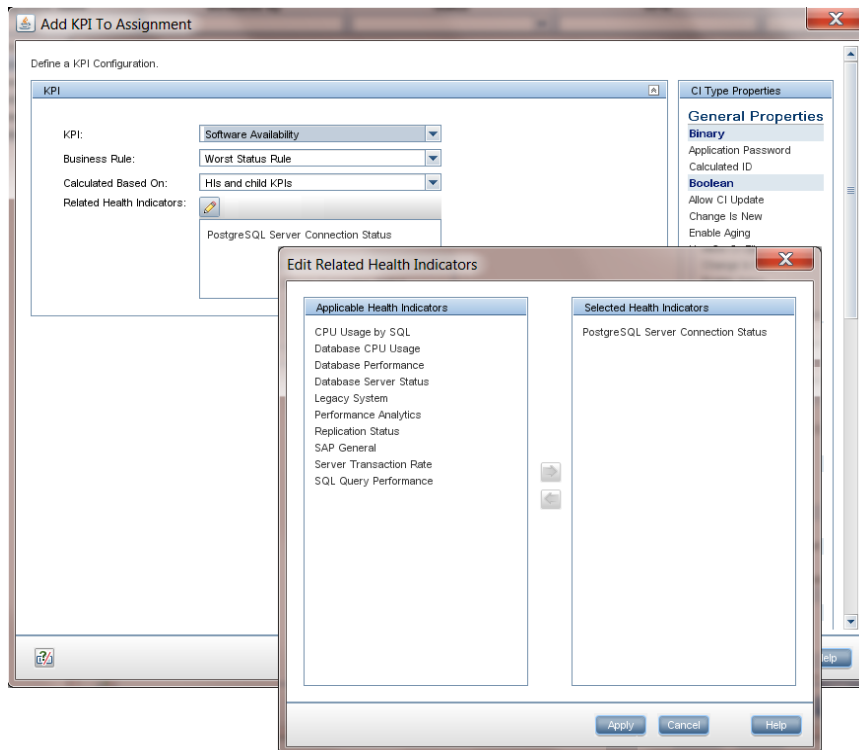
1. Click **Admin > Service Health**.
2. In the resulting window, click **Assignments** tab.
3. Select the CIT from the tree view for the HIs you want to assign to KPIs.
4. Click  **New**.
5. In Add KPI Assignment for CI Type: PostgreSQL Server window, provide a Name and a Description for KPI assignment.
6. Click  **New** under KPI Configuration.



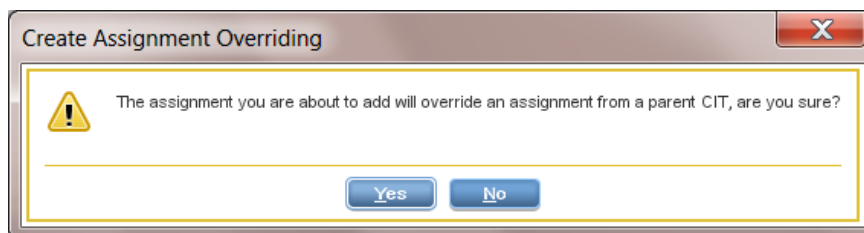
7. In the Add KPI To Assignment window, if you assign availability HIs, then select **Software Availability** for KPI.

Note: Retain the default values for Business Rule and Calculated Based On fields.

8. Click  **New** to select Related Health Indicators.
9. Click  in the Related Health Indicators field, and then select the appropriate health indicators to assign to selected KPI.
10. Click **Apply**.
11. Click **Save** to save the assignments.
12. To assign the performance HIs to the Software Performance KPI, repeat the steps 7 to 11.



While saving the HIs, if there are KPI assignments already available through inheritance from the parent CIT, you will see the following warning message, click **Yes**.



Creating Monitoring Components

To create Monitoring Components, follow these tasks:

Task 1: Importing instrumentation

Instrumentation that performs discovery, metrics collection, and other value additions must be imported into OMi so that it can be assigned to the appropriate components. When those components are deployed, appropriate instrumentation are deployed automatically.

OMi does not support uploading instrumentation using BSM User Interface (UI). Instrumentations are stored as Zip in the Database and are not available on the file system. Hence, you need to create a folder structure and then copy the files into appropriate folders, Zip the folder, and upload it. Following command and steps guides you through the process:

1. Create a sample instrumentation directory

On OMi, Configexchange tool can create a sample folder structure for instrumentation. Following command creates a sample folder:

```
C:\HPBSM\opr\bin\ConfigExchange.bat -cin -o "c:\postgresql\instr"
```

A folder structure is created as shown:



2. Copy the instrumentation to appropriate folders

Copy the instrumentation pertaining to OS platforms into appropriate folders. If some files are common across various OS versions, then copy them at the higher level. For example, PostgreSQL_Discovery.pl is the discovery script that discovers PostgreSQL CIs in all platforms. So, it needs to be copied under the directory `instr`. If you have instrumentation that is common to all Windows platforms, then you can copy instrumentation into Windows directory.

3. Import instrumentation into OMi

After the instrumentation files are copied to appropriate folders, it needs to be imported into OMi. Following command zips the instrumentation folder `c:\postgresql\instr` and imports it as an instrumentation category PostgreSQL:


```
c:\HPBSM\opr\bin\ConfigExchange.bat -username <username> -password <password>  
-inn PostgreSQL -ul -i "c:\postgresql\instr"
```

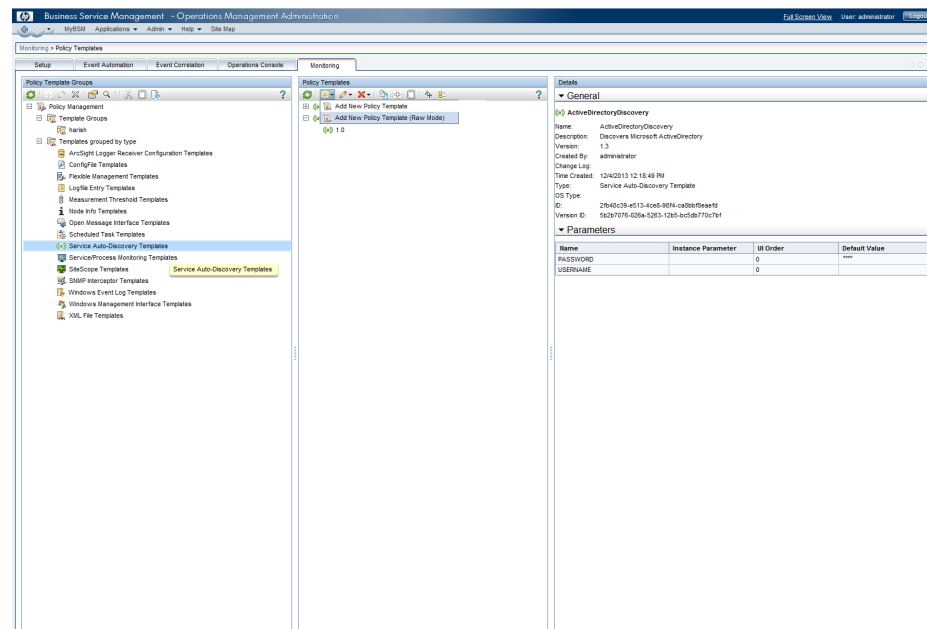
You can list all the instrumentations that are uploaded into the OMi server by using the following command:

```
c:\HPBSM\opr\bin\ConfigExchange.bat -username admin -password admin -l -inst  
runname ALL
```

Task 2: Creating discovery policy

Before proceeding with this step ensure RTSM model for the application is already created in the RTSM. To create a discovery policy, follow these steps:

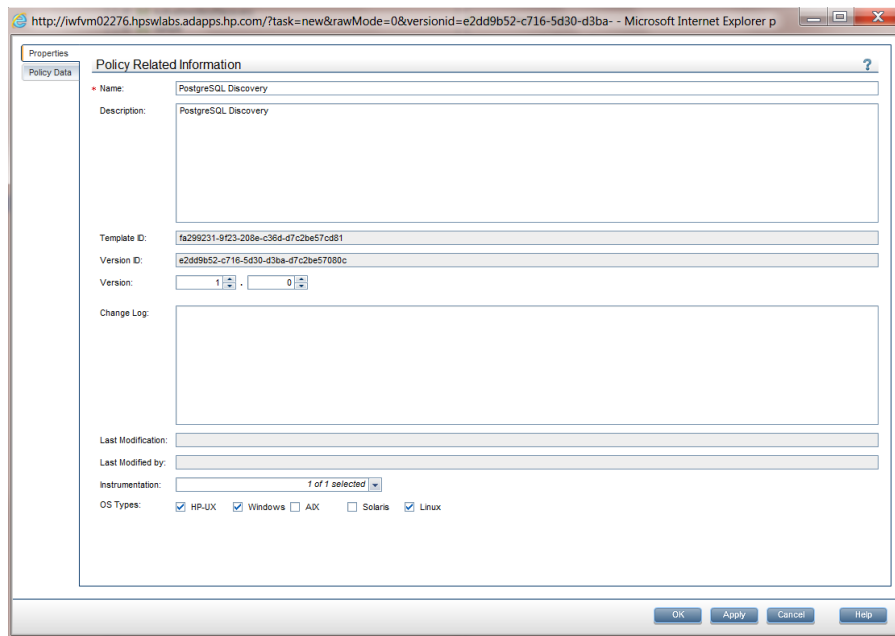
1. Navigate to **Operations Manager > Policy Templates**.
2. Click **Admin > Operations Management > Monitoring > Policy Templates**.
3. Click **Service Auto-Discovery Templates**.
4. Click  **New**, and then select **Add New Policy Template**.



5. Provide the Name and Description in the Policy Related Information window.
6. Select **Custom Discovery** in the Instrumentation field.
7. Select the **OS Types** from the list.

Note: You need to select the platform on which the discovery policy is deployed. PostgreSQL runs on Linux, UNIX, and Windows. Hence, those OS types are selected.

8. Select **Policy data**.



9. In the resulting window, in the Policy Raw Data field, copy and paste the whole content of the file in the [Appendix](#).

- Alternatively, you can create a discovery policy in OMW/L/S and export the policy and import into OMi.

For more information about discovery policies, see the following location in the *BSM Online Help*:

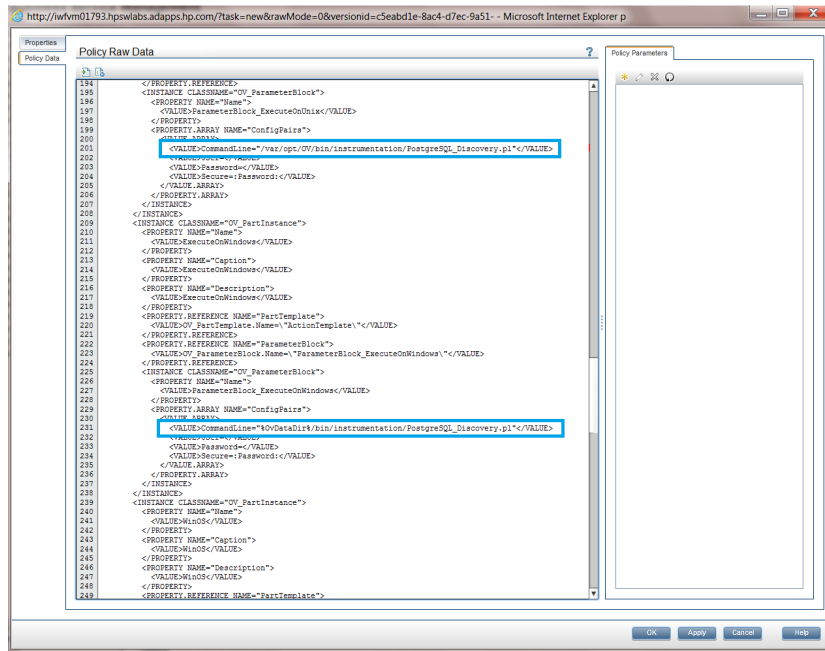
Application Administration > Operations Management > Monitoring > Policy Templates > Configuring Service Auto-Discovery Policies.

10. Create a script or binary file that discovers the application CIs along with the related relationships and create an XML file with the discovered data. Update the name of the script or binary file in the XML that you copied into Policy Raw Data. If the application runs on any Unix or Linux distribution and the discovery script or Binary is meant for Unix or Linux, then update name of the file in the first location marked in the following screen shot. In the example, **PostgreSQL_Discovery.pl** is updated.

- XML that is generated by this script need to adhere to a structure. For more information about XML schema definition and XML element descriptions, see the following location in the *BSM Online Help*:

Application Administration > Operations Management > Monitoring > Policy Templates > Configuring Service Auto-Discovery Policies.

11. If the application runs on any Windows flavors and the discovery script or binary is meant for Windows, then update name of the file in the second location marked in the following screen shot.



12. Save the discovery policy.

Task 3: Creating Policy Templates and parameters

In this section we create the policies that were identified in the [Identifying Policy Templates](#) section and set the CI hints and appropriate Custom Message Attributes (CMAs) in each policy so that events generated by these policies reaches appropriate CIs and ETI or HI combination.

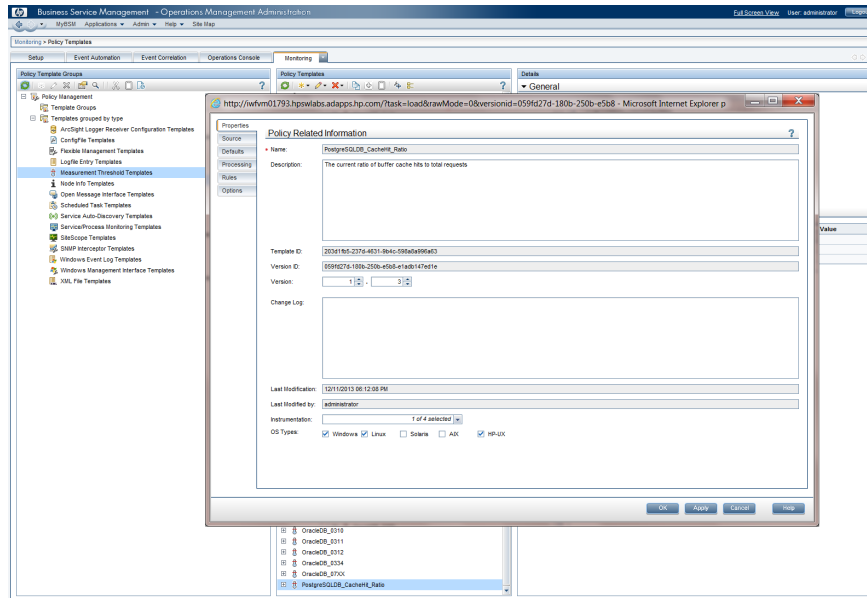
Let us create PostgreSQLDB_CacheHit_Ratio policy which is previously identified.

To create a Policy Template, follow these steps:

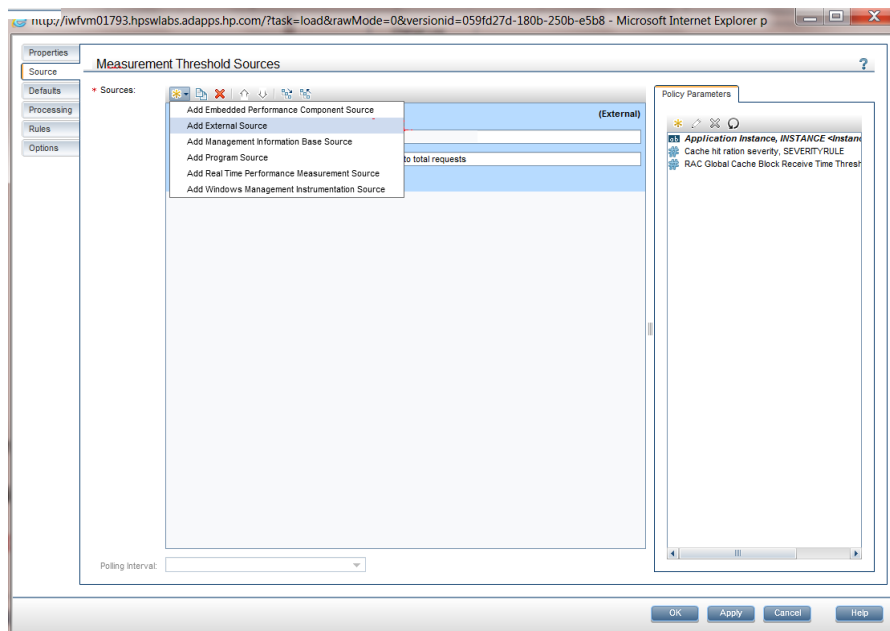
1. Navigate to the Policy Templates window.
2. Click **Admin > Operations Management > Policy Templates**.
3. To create a policy of a given type, click the policy type under Templates grouped by type.

In the example, it is Measurement Threshold Policy.

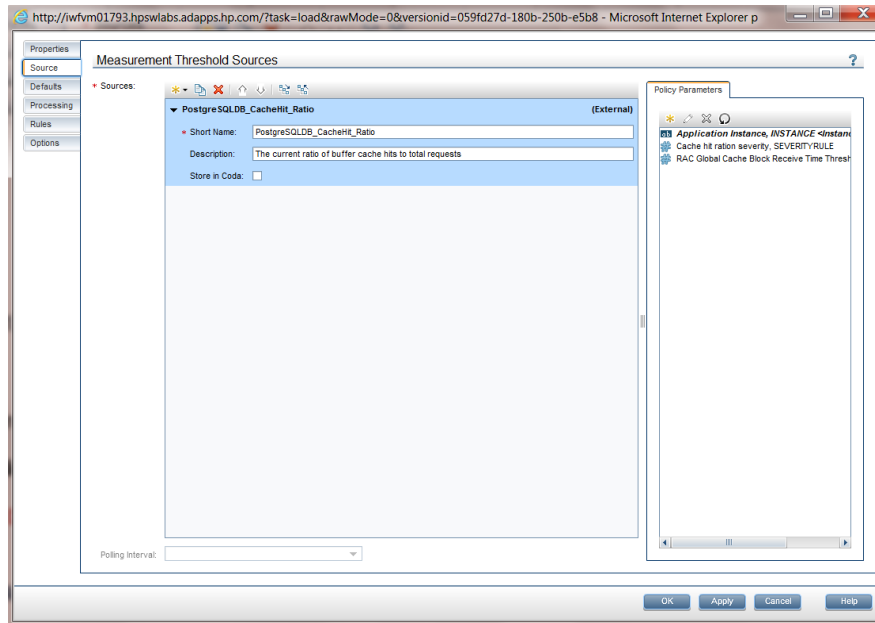
4. Provide a Name and Description for the policy.
5. Select the instrumentation for the Policy Template.
6. Select the OS types on which this policy needs to be deployed. Select the OS platforms that supports PostgreSQL.



7. Click the **Source** tab.
8. Select **Add External Source**. (This policy gets the values from an external source (Instrumentation))

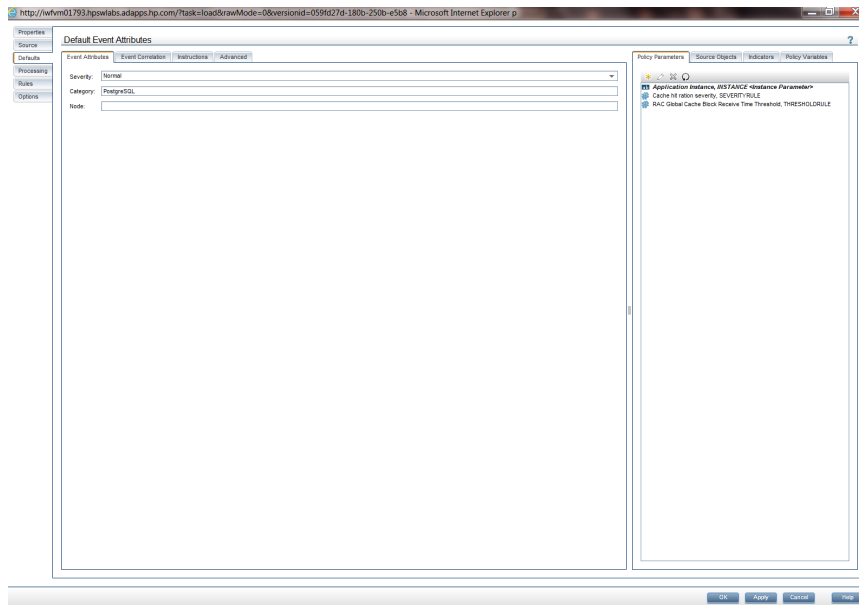



9. Provide a Short Name and Description in the Measurement Threshold Sources window.

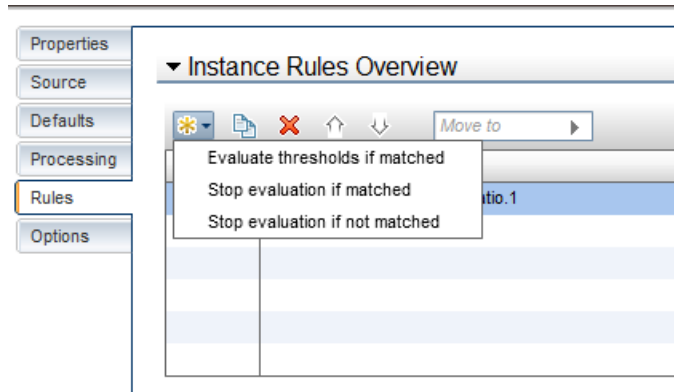


10. Provide the following default values in the Defaults tab:

- Provide a default value for the events that are generated from this policy. This value is used when there are no severity mentioned for the events in the Rules tab.
- Provide the name of the logical group to which the event belongs to (For example, Database, Security, or Network). The event category is similar to the HP Operations Manager message group.

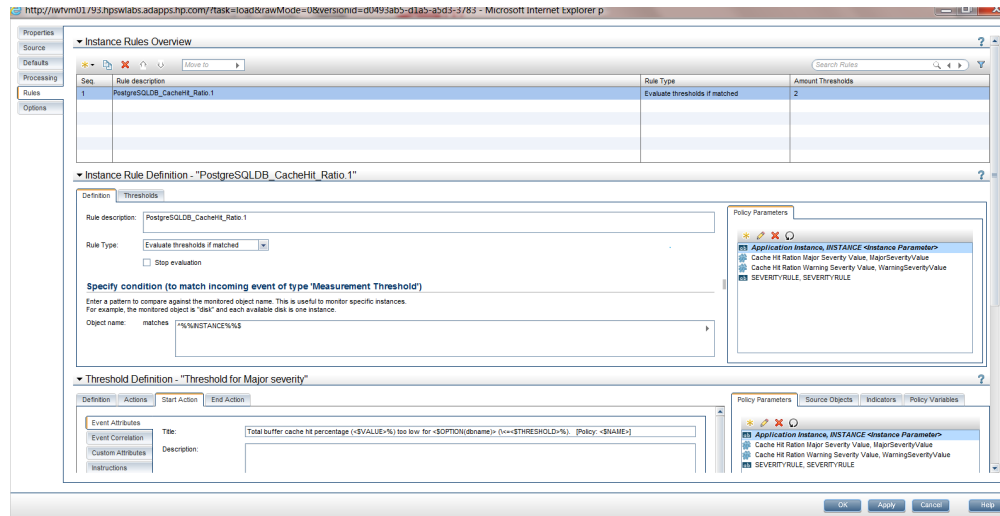


11. Create new rules in the Rules tab. To create a new rule, follow these steps:
 - a. Click  **New**, and then select **Evaluate thresholds if matched** from the list.
 - b. Provide a rule description.



12. Define an Instance rule definition. To define an Instance rule definition, follow these steps:
 - a. Provide a rule description.
 - b. Select a rule type.
 - c. Create a condition for matching monitored object name. In the example, a parameter **INSTANCE** has been introduced for the condition. You can specify a parameter by prefixing and suffixing a variable name with **%%**. For example, **%%ParameterName%%**.
 - d. Modify the parameter from the Policy Parameters tab and make it as an instance parameter by selecting the Instance Parameter check box.

An instance parameter enables you to create Policy Templates that monitors multiple instances of the same type of object (For example, Multiple instances of PostgreSQLDB). Each Policy Template can have only one instance parameter. When you add an instance parameter to a Policy Template, all other parameters become dependent on it. The user can specify separate values for the dependent parameters of each instance. Not all policy types support instance parameters. For more information, see the *BSM Online Help*.

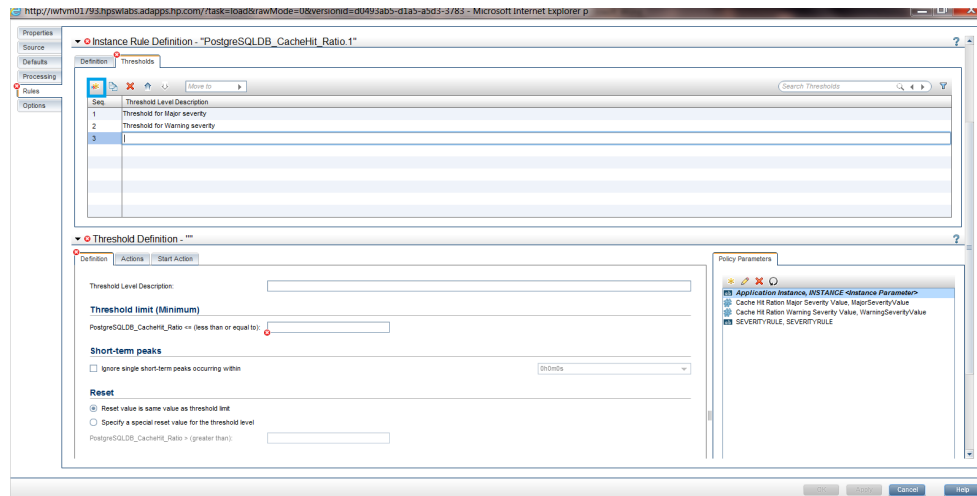


13. Add thresholds to Policy Template. To add thresholds to a Policy Template, follow these steps:

- a. Click **Thresholds** tab, and then click  **New**.

A new threshold is added.

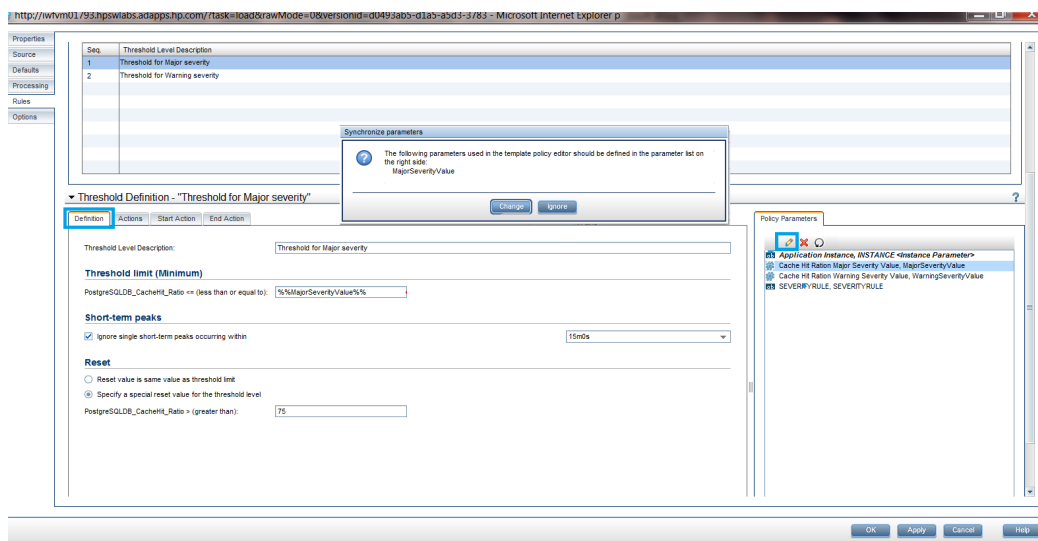
- b. Provide a description for the threshold.




14. Provide threshold definitions. To provide threshold definitions, follow these steps:

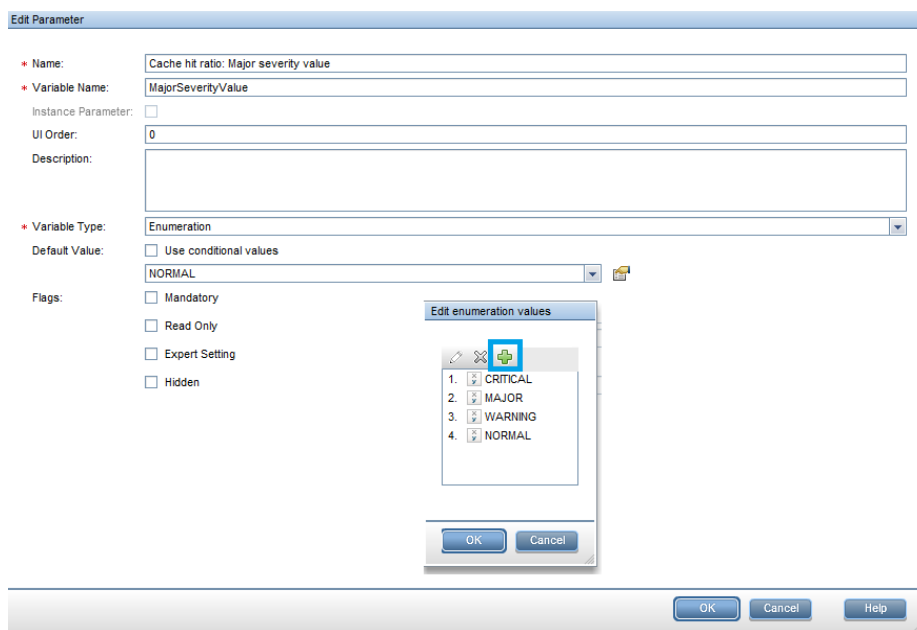
- a. Provide a meaningful threshold description in the Threshold Level Description field. In the example; a threshold has been created for Cache hit ratio major severity.

- b. Provide a threshold limit in the Threshold limit (Minimum) field. In the example; the Major severity has been parameterized.
- c. If you want to ignore any single short time peaks, select the Short-term peaks check box and provide the time. Since it may not be reasonable to create an event when a threshold is exceeded only for a short time, you can define a minimum time period over which the monitored value must exceed the threshold before generating an event. For an event to be sent, the value must be greater than the threshold each time the value is measured during a duration that you select.
- d. The reset value is a limit below which the monitored value must drop (or exceed, for minimum thresholds) to return the status of the monitored object to normal. After the status of a monitored object returns to normal, a new start event can be issued if the monitored value again crosses the threshold value. You can either use the same value as the threshold limit, or specify a different reset value. In the example, when the Cache hit ration goes high to 75, then the value will be reset to normal.
- e. Click **Synchronize Parameters** to check the Policy Template to make sure that the variables in the format %%<variable_name>%% have corresponding parameters. Each variable must have one corresponding parameter. It checks for the parameters that are not used and for which corresponding variable does not exists in the Policy Template.
- f. If you have added a new parameter, then get confirmation questions to change the parameter list, click **Change** to confirm.



15. Edit the parameter. To modify a parameter, follow these steps:

- a. Click **Edit Parameter** on Policy Parameters Tab.
- b. In the Edit Parameter window, provide a display name, that is self-descriptive which can be understood by the user when he wants to provide values for the parameters.
- c. Select the position of this parameter in the list of parameters.
- d. Define the type of value that consumers can specify for the parameter. In the example, variable type is selected as Enumeration. Select the Use conditional values check box to add a list of conditional default values. You can configure conditional default values based on the type of operating system of the host node to which the Policy Template is deployed.
- e. To specify the enumeration values, click  **Edit Enumeration Values**.
- f. Configure the values using the Edit Enumeration Values window. You can also change the order of the values in this window.



The screenshot shows the 'Edit Parameter' window with the following fields and settings:

- Name:** Cache hit ratio: Major severity value
- Variable Name:** MajorSeverityValue
- Instance Parameter:** ☐
- UI Order:** 0
- Description:** (empty text area)
- Variable Type:** Enumeration
- Default Value:** NORMAL
- Flags:**
 - ☒ Use conditional values
 - ☐ Mandatory
 - ☐ Read Only
 - ☐ Expert Setting
 - ☐ Hidden

The 'Edit enumeration values' dialog is open, showing a list of values with up/down arrows for reordering:

1. CRITICAL
2. MAJOR
3. WARNING
4. NORMAL

Buttons: OK, Cancel, Help.

16. Define Actions.

You can choose the type of actions you want to perform by following the guidelines:

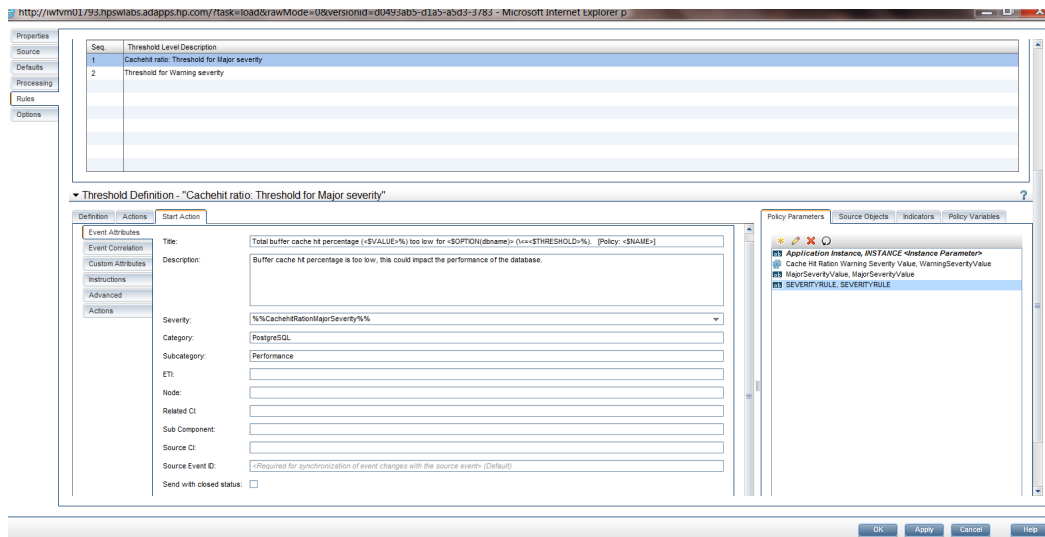
- When first time a threshold is met (Start Actions)
- During every polling interval if the start action of the rule was carried out at a previous polling interval and the reset value is not reached. (Continue Actions)
- End actions are carried out after the threshold crosses the reset value only if the start action for that rule was carried out. If the value drops below two thresholds within one polling

interval, the end actions of the lowest rule that performed start actions are carried out (End Actions)

▼ Threshold Definition - "Cachehit ratio: Threshold for Major severity"

Definition	Actions	Start Action	End Action
<hr/> Start Actions <hr/> <p>If the threshold limit is met or crossed, then start the specified actions and stop checking other threshold levels.</p> <p>Edit the 'Start Actions' event</p> <hr/> Continue Actions <hr/> <p>If the threshold limit is still met or crossed, then start the specified actions and stop checking other threshold levels.</p> <p><input type="checkbox"/> Define special 'Continue Actions'</p> <p>Edit the 'Continue Actions' event</p> <hr/> End Actions <hr/> <p>Start these actions if the last action for this threshold level was 'start' or 'continue', and no threshold level is met or crossed.</p> <p><input checked="" type="checkbox"/> Start the specified 'End Actions'</p> <p>Edit the 'End Actions' event</p>			

17. Define a start action. To define a start action, follow these steps:
 - a. Click **Start Action**; provide a brief description of the event with relevant information about the metric value that has crossed threshold.
 - b. Provide a description about the Event. Provide more information about the event to the user to solve the problem.
 - c. Provide the severity of the event that enables the user to estimate the situation and provide a correct attention to the event. In the example, severity has been parameterized.



18. Provide Event Correlation keys and CI/ETI hints.

- This key is an identifier used to identify duplicates and for Close Events with Key.
- Key for closing the events, if events with the event key that you type here exist in the Operations Management event database when this event is received, these events are automatically closed. You can use pattern matching and policy variables available in the policy variable tab to match multiple event keys.
- Provide a CI hint, which the BSM uses to identify the CI related to the event. Use the following format:

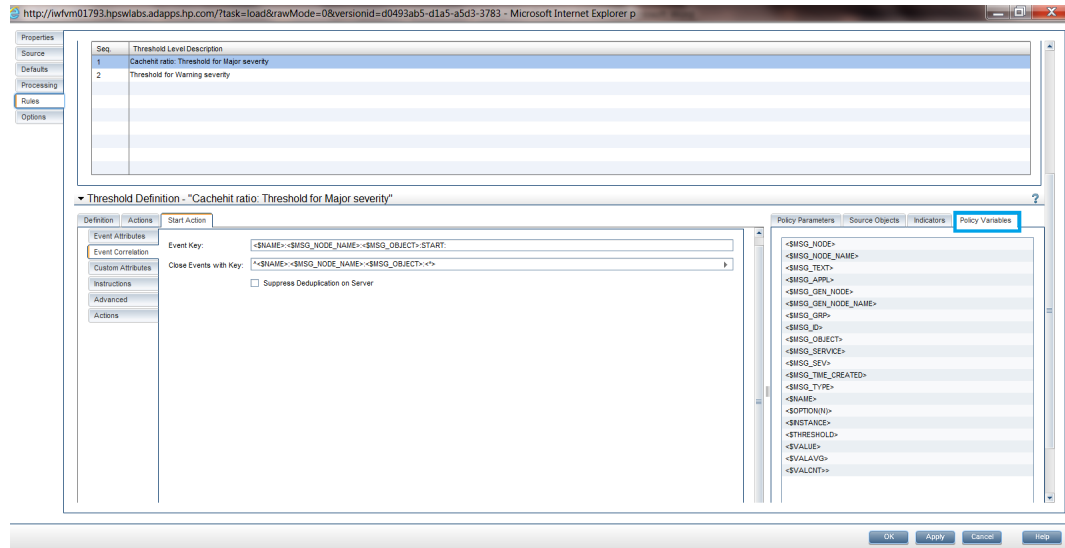
<hint 1>:<hint 2>....<hint n>@@<hostname>

For example, PostgreSQLDbId01@@node.example.com or C:@@server.example.com

- In the ETI field, provide an ETI resolution hint, which the BSM uses to associate the event with an ETI and for event correlation. Use the following format:

<ETI name>:<ETI state>:<metric value>

For example, <PostgreSQLCacheHitRatio>:<Low>:<40>



19. In the Instructions tab, provide more information on probable cause, potential impact of the issue and suggested actions to help the user to solve the situation.

For more information about other attributes on the Policy Template, see the following location in the *BSM Online Help*:

Application Administration > Operations Management > Monitoring > Policy Templates > Configuring Measurement Threshold Policies

Similar attributes can be set in continuous and end actions tabs.


For more information about types of Policy Templates, see the following location in the *BSM Online Help*:

Application Administration > Operations Management > Monitoring > Policy Templates



Task 4: Creating Aspects

This section guides you through to create Aspects that you have identified in the chapter [Identifying monitoring components](#).



To access Aspects, follow steps:

1. Navigate to Management Templates & Aspects page.
2. Click **Admin > Operations Management > Management Templates & Aspects**.
3. Create configuration folder. To create a new configuration folder, follow these steps:
 - a. Click  **New** in the Configurations Folder pane, and provide a Name and Description for the configuration folder.

For PostgreSQL database, a configuration folder PostgreSQL has been created under Database.


- b. Under PostgreSQL, you can create a folder each for the Aspects and the Management Templates. For Example, PostgreSQL Aspects and PostgreSQL Management Templates are the folders created for PostgreSQL.
4. Create an Aspect. To create a new Aspect, follow these steps:
 - a. Click  **New** in the Management Templates & Aspects pane, and select  Create Aspect.

The Add New Aspect window appears.
 - b. In the Add New Aspect window, provide the Aspect details such as Name and Description.
5. Associate Aspects with the CIT. To associate Aspects with the CIT, follow these steps:
 - a. Click **Next** in aspect creation wizard.

CIT tab appears.
 - b. Select the CIT that you want to associate the Aspect with from Available CI Types list.
 - c. Move the CIT from Available CIT to Assigned CIT using  tab.
 - d. Select **Node compatible** for Aspects with discovery policies.
6. Add instrumentation category. To add instrumentation category, follow these steps:
 - a. Navigate to Instrumentation tab by clicking the **Next**.
 - b. In the Instrumentation step, click  **Add Instrumentation** to add instrumentation to the Aspect.

The Add Instrumentation window appears, which enables you to select the instrumentation that you want to add.
 - c. Click **OK**.

Instrumentation category is added.
7. Add Aspects.

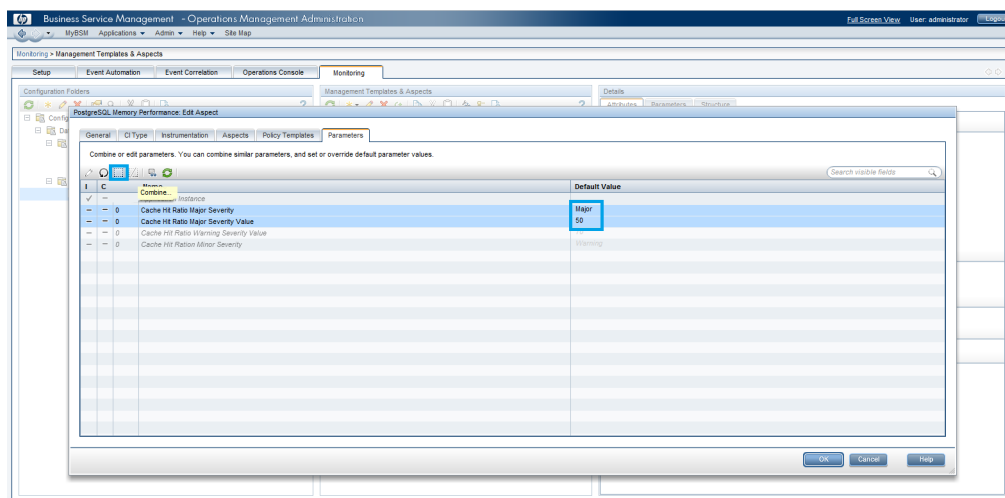
This is an optional step if you want to add an existing Aspect as part of the Aspect that you are creating.
8. Add Policy Templates. To Add Policy Templates, follow these steps:
 - a. Navigate to Policy Templates tab by clicking the **Next**.
 - b. Click  **Add Policy Template** to assign a Policy Template.

The Add Policy Template to Aspect window appears, listing all Policy Templates installed on the system.

- c. Select all the Policy Templates you want to assign to the Aspect to include Policy Templates created earlier.
 - d. Click **OK**.
9. Combine or Edit parameters.

This is an optional step from the parameters tab.

- You can override the default value of the parameters defined in the Policy Templates that you added in the Add Policy Templates tab. You can set the value of a parameter at Aspect level.
- You can combine parameters. When there are similar parameters from different templates that need to have same value from the user, then they can be combined. For Example, Username and Password.





For more information about Aspects, see the following location in the *BSM Online Help*:

Application Administration > Operations Management > Monitoring > Management Templates and Aspects > Configuring Aspects

Task 5: Creating Management Templates

Provide the general information about the Management Template.

To create a new Management Template, follow these steps:

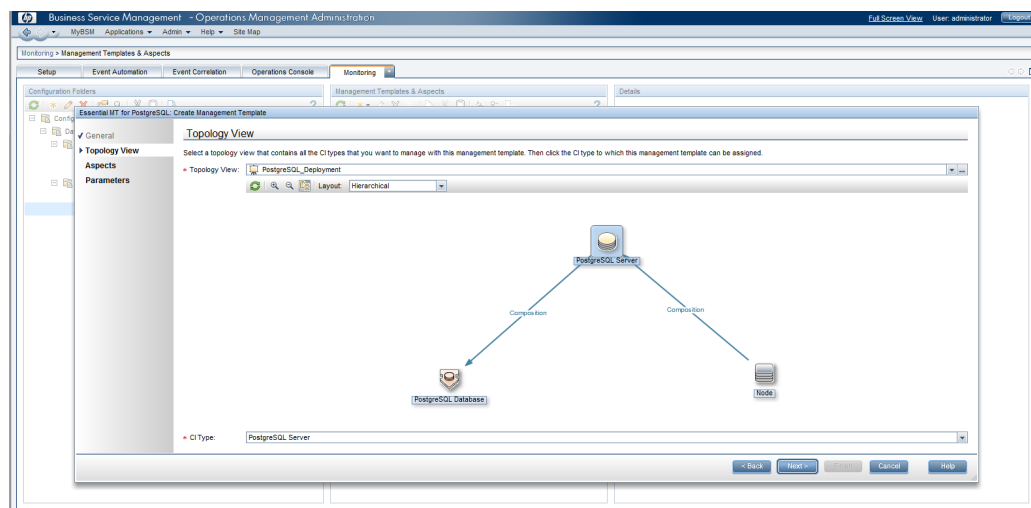
1. Click  **New** in the Management Templates & Aspects pane, and then select  **Create Management Template**.


The Add New Management Template window appears.

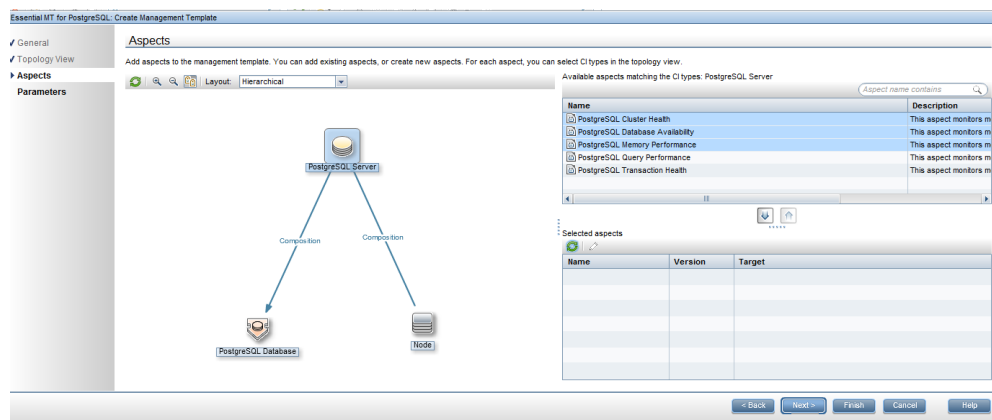
2. Provide a name for the Management Template.
3. Provide description about the Management Template, and then click **Next**.
4. Assign a topology to the Management Template. Follow these steps:
 - a. Navigate to Topology View.
 - b. Select the appropriate topology view from the Topology View drop-down box.

After the View is selected, the view is rendered in the pane.

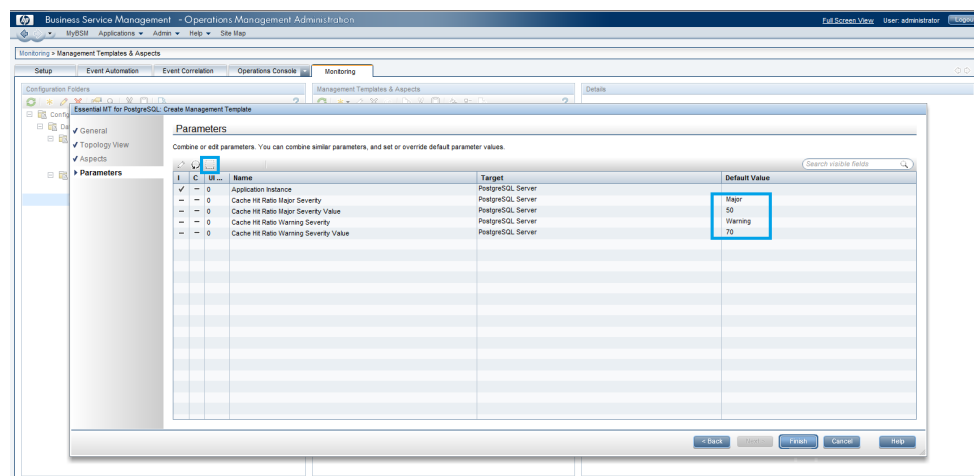
- c. Select the key CI Type that you want to assign to Management Template from the drop-down box.



- d. Click **Next**.
5. Assign Aspects to CIT. To assign Aspects to CIs, follow these steps:
 - a. Select the CIT from the View.
 - b. Aspects associated to that CI are listed in the list box, select the aspects that needs to be deployed to the CI during the deployment.
 - c. Move the selected Aspects using  tab.
 - d. Click **Next**.



6. Set the Management Template Parameters.
 - a. In the Parameters window, you can select similar parameters across Aspects and combine them into one at Management Template level.
 - b. You can provide default values for the parameters.
 - c. Click **Finish** to save the Management Template.



Creating an RTSM Package

The model, the View, and the query that you have created are not specific to any monitoring software. It caters to various different monitoring solution needs, hence, these components need to be separately selected and exported as an RTSM package. This section guides you through how to create and export an RTSM package with relevant components. This is a two-step process, first you need to create an RTSM package with components and then export it so that, it can be shipped for consumption. Follow these tasks for packaging an RTSM Package:


Task 1: Accessing manager

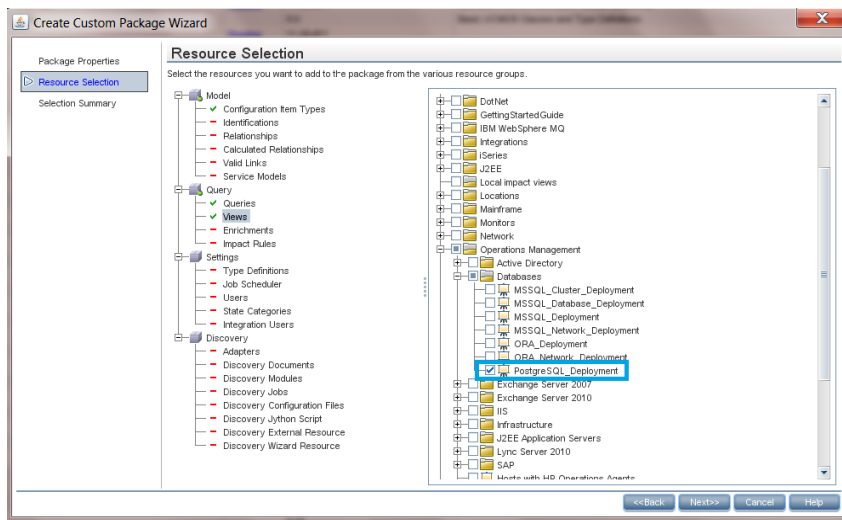
To access the manager, follow these steps:

1. Navigate to the package manger.
2. Click **Admin > RTSM Administration > Administration > Package Manager**.

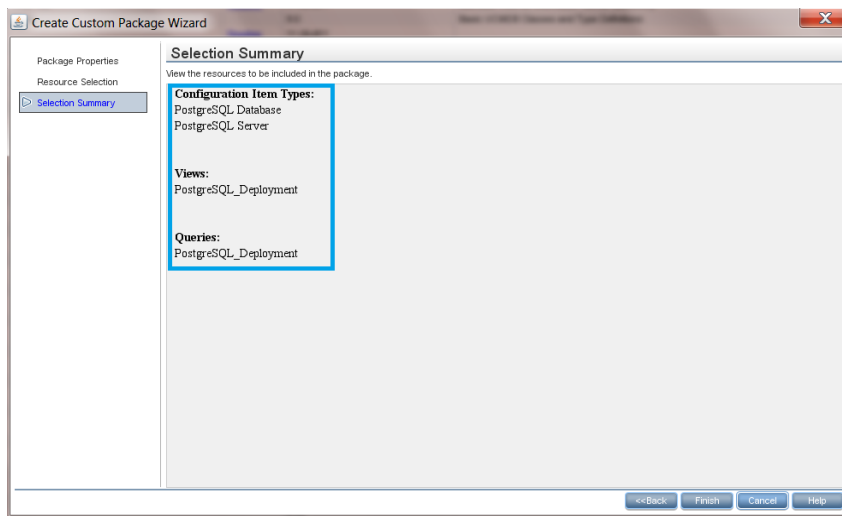
Task 2: Creating an RTSM package

To create an RTSM package follow these steps:

1. Click  **New**.
2. In the Create Custom Package Wizard window, provide Package Name, Package Category, and Package Description.
3. Click **Next**. In the Resource Selection window, select the CITs, the View, and the Query that you have created, and then click **Next**.



4. In the Selection Summary window, you can see the components that are selected. In the example, PostgreSQL CITs, View, and query have been selected. Verify and click **Finish**.



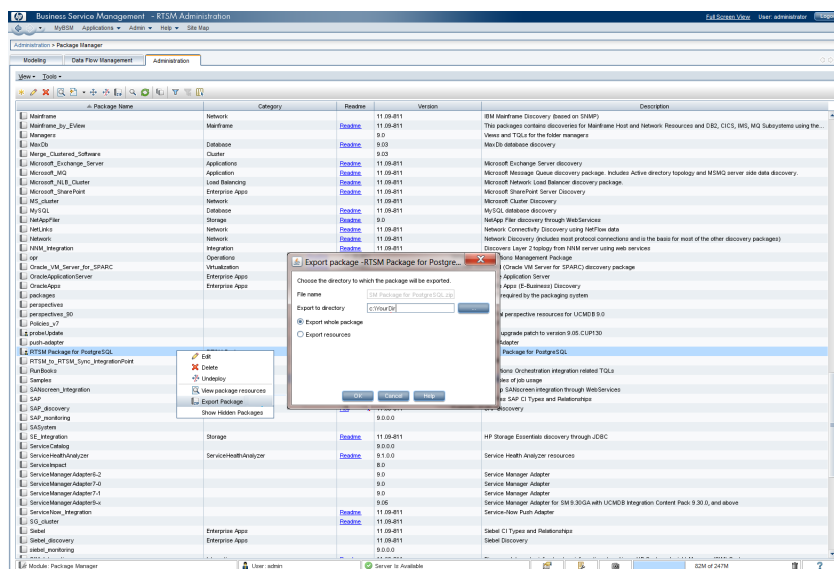
Task 3: Exporting an RTSM package

In the previous task, you created an RTSM package. Now you must export the RTSM package.

To export, follow these steps:

1. Navigate to the package manger.
2. Click **Admin > RTSM Administration > Administration > Package Manager**.
3. Select and right-click the name of the package you want to export, and then select **Export Package**.
4. Provide the directory where the package needs to be exported.
5. Click **OK**.

Package will be exported in .zip format (For example, RTSM Package for PostgreSQL.zip).



Packaging OMi MP

This sections provides information about creating and downloading OMi MP. To package an OMi MP, follow these tasks:

Task 1: Accessing content manager

To access the content manager, follow these steps:

1. Navigate to **Admin > Operations Manager > Content Packs**.
2. In the New Content Pack creation window, provide the MP details such as Display Name, Name, Version, and Description.

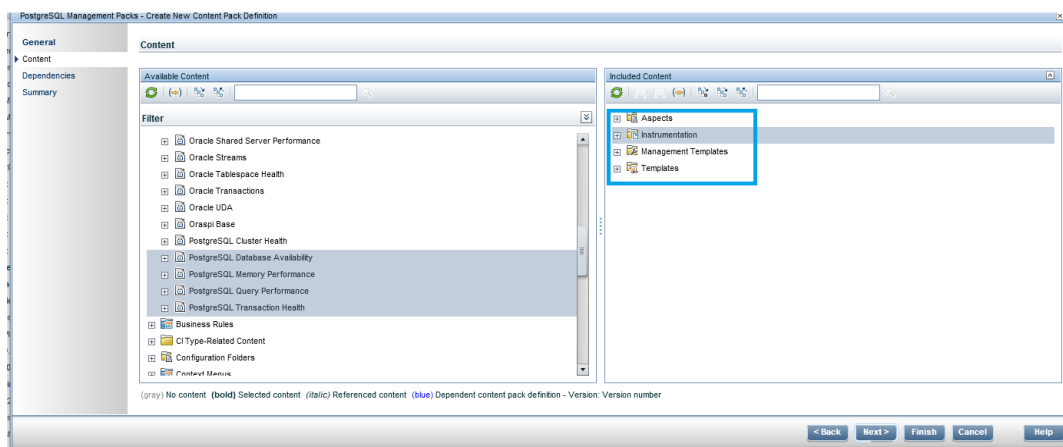
Task 2: Packaging OMi MP

To select the components to be exported in the OMi MP, follow these steps:

1. Select artifacts such as ETIs, HIs, Management Templates, Aspects, instrumentation, Policy Templates to be exported into OMi MP.

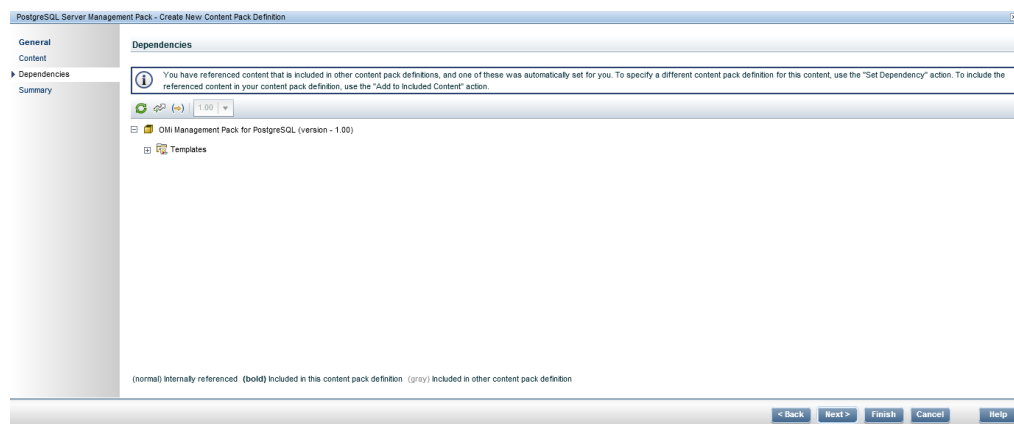
Note: You need to select all the components.

2. Move the selected components from Available Content pane to Included Content pane.
3. Click **Next**.



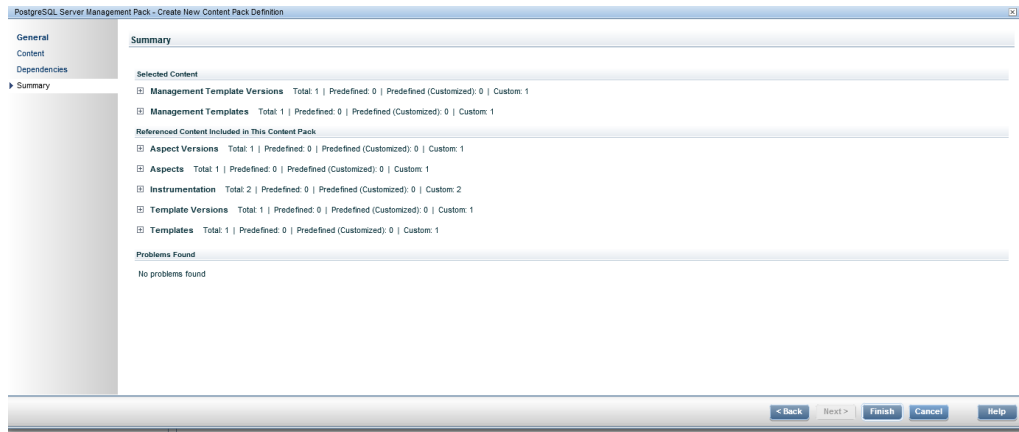
Task 3: Setting Dependencies

1. In the resulting Dependencies screen, the components referred from other OMi MPs are listed.
2. You can select the components referred from other OMi MPs to be a part of your OMi MP or you can make your OMi MP dependent on the other OMi MP. In the second case, the other OMi MP needs to be installed before the installation of dependent OMi MP.




3. Summary page lists the components that are selected; you can verify and click **Finish** to

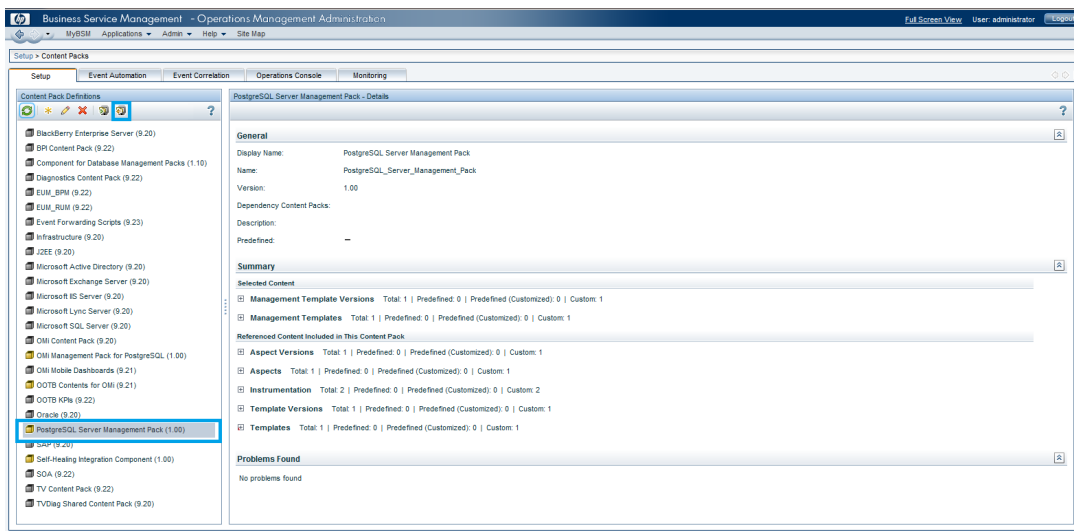
create an OMi MP.



Task 4: Exporting OMi MP

To export an OMi MP, follow these steps:

1. Select the OMi MP to be exported from the content pack definitions list.
2. Click  **Content Pack Definitions and Content**.
3. In the resulting window, select the directory and provide a name for the ZIP file, and then click **Save**.



Validating OMi MP

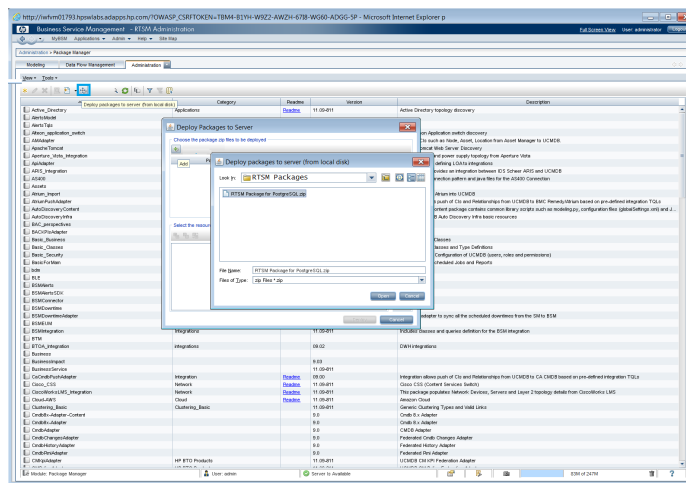
This section provides information about installing and testing an OMi MP.

Task 1: Importing RTSM package

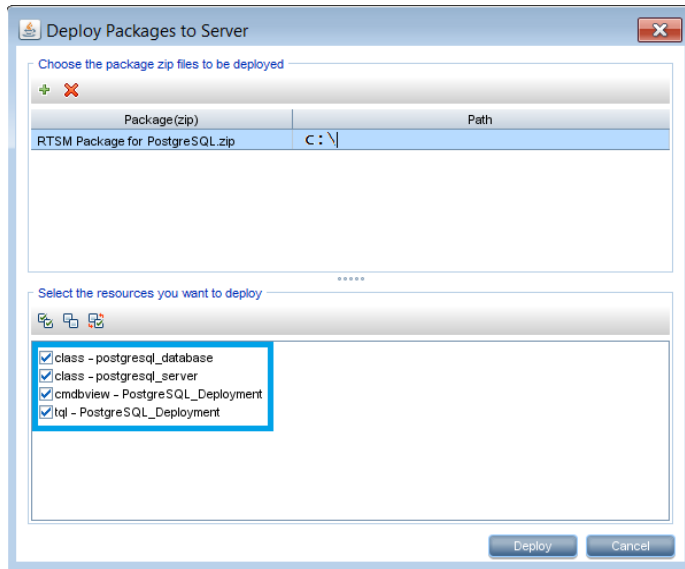
For importing an RTSM package, you need to specify the download location of the RTSM package.

To import an RTSM package, follow these steps:

1. Navigate to the package manger.
2. Click **Admin > RTSM Administration > Administration > Package Manager**.
3. Select **Deploy packages to server (From local disk)**.
4. Click **Add** in the Deploy packages to Server window.
5. Select the RTSM package from the directory.
6. Click **Open**.




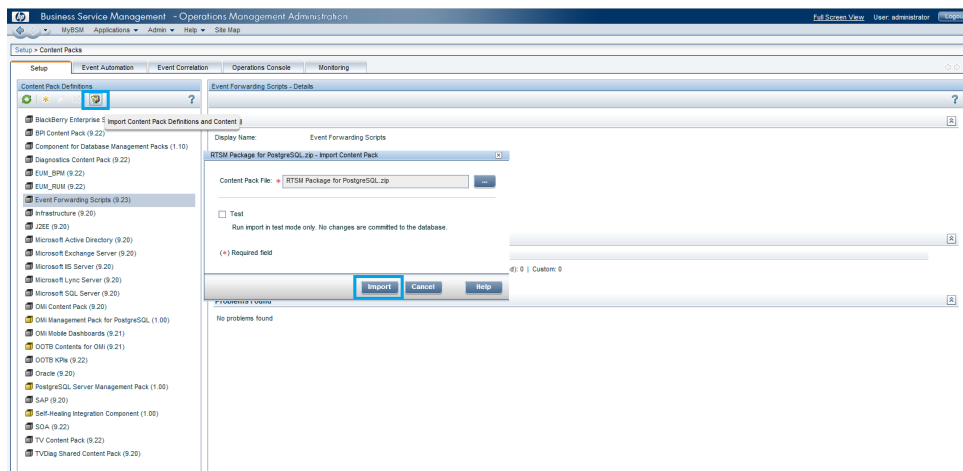
7. In the Deploy Packages to Server window, select the RTSM package you want to import.
8. Select the resources to be deployed.
9. Click **Deploy**.



Task 2: Importing OMi MP

To import an OMi MP, follow these steps:

1. Navigate to content manager.
2. Click **Admin > Operations Manager > Content Packs**.
3. Click  **Import Content Pack Definitions and Content**.
4. In the Import Content Pack window, select the OMi MP Zip file to be imported.
5. Click **Import**.



Testing

The following section provides information about the high-level guidelines for testing an OMi MP. You can create an in depth test plan with scripts and test cases to test an OMi MP.

After importing an RTSM package and an OMi MP package into the test environment, you can begin testing the environment. To ensure the quality of an OMi MP, you can consider the following guidelines:

1. Consider a testing environment that depicts the actual customer scenario, and test it with possible deployment of applications.
2. Test the following:
 - a. Check whether you are able to discover CIs.
 - b. Check whether appropriate parameters are appearing with the default parameters. The mandatory parameters must be filled by the user.
 - c. Check whether the deployment of a Management Template to a View deploys appropriate Aspects to the corresponding CIs.
 - d. Test for each metric that is getting monitored for all the possible events and acknowledgments.
 - e. Check whether the events are reaching corresponding CIs and appropriate HIs and ETIs.
 - f. Check whether the events are generated when the thresholds exceed.
 - g. Check whether the correct View appears when an event is selected.

Appendix: Discovery

```
<AutoDiscPolicy progVERSION="X.05.00">
  <Version>1.0</Version>
  <Discover>
    <ManagedModules>
      <ManagementModule name="customdiscovery">
        <Caption>Custom-Discovery</Caption>
        <Description>Module supporting custom discovery scripts</Description>
        <RootStd>DiscoveredElement</RootStd>
      </ManagementModule>
    </ManagedModules>
  </Discover>
  <Schedule>
    <Options>
      <Task>Daily</Task>
      <Timesel>Exactly</Timesel>
    </Options>
    <Time>
      <Minute>0</Minute>
      <Hour>2</Hour>
      <Month />
      <Year />
      <DofW />
      <DofM />
    </Time>
  </Schedule>
  <ManagementModuleElements NAME="customdiscovery">
    <INSTANCE CLASSNAME="OV_ManagementModule">
      <PROPERTY NAME="Name">
        <VALUE>customdiscovery</VALUE>
      </PROPERTY>
    </INSTANCE>
  </ManagementModuleElements>
</AutoDiscPolicy>
```

```
<PROPERTY NAME="Caption">
<VALUE>Custom-Discovery</VALUE>
</PROPERTY>
<PROPERTY NAME="Description">
<VALUE>Module supporting custom discovery scripts</VALUE>
</PROPERTY>
<PROPERTY NAME="RootServiceTypeId">
<VALUE>DiscoveredElement</VALUE>
</PROPERTY>
</INSTANCE>
<INSTANCE CLASSNAME="OV_ServiceTypeDefinition">
<PROPERTY NAME="GUID">
<VALUE>DiscoveredElement</VALUE>
</PROPERTY>
<PROPERTY NAME="Caption">
<VALUE>Discovered Element</VALUE>
</PROPERTY>
<PROPERTY NAME="Description">
<VALUE>service type used by custom discovery scripts</VALUE>
</PROPERTY>
<PROPERTY NAME="Name">
<VALUE>DiscoveredElement</VALUE>
</PROPERTY>
<PROPERTY NAME="CaptionFormat">
<VALUE>$caption$</VALUE>
</PROPERTY>
<PROPERTY NAME="DescriptionFormat">
<VALUE>$description$</VALUE>
</PROPERTY>
<PROPERTY NAME="KeyFormat">
<VALUE>$key$@@</VALUE>
</PROPERTY>
<PROPERTY NAME="CalcRuleId">
```

```
<VALUE>DDK_DefaultCalculationRule</VALUE>
</PROPERTY>
<PROPERTY NAME="MsgPropRuleId">
<VALUE>DDK_DefaultPropagationRule</VALUE>
</PROPERTY>
<PROPERTY NAME="ManagementModuleId">
<VALUE>customdiscovery</VALUE>
</PROPERTY>
</INSTANCE>
<INSTANCE CLASSNAME="OV_ServiceTypeComponent">
<PROPERTY.REFERENCE NAME="GroupComponent">
<VALUE>OV_ServiceTypeDefinition.GUID="\DiscoveredElement\"</VALUE>
</PROPERTY.REFERENCE>
<PROPERTY.REFERENCE NAME="PartComponent">
<VALUE>OV_ServiceTypeDefinition.GUID="\DiscoveredElement\"</VALUE>
</PROPERTY.REFERENCE>
<PROPERTY NAME="PropRuleId">
<VALUE>DDK_DefaultPropagationRule</VALUE>
</PROPERTY>
</INSTANCE>
<INSTANCE CLASSNAME="OV_ServiceTypeDependency">
<PROPERTY.REFERENCE NAME="Dependent">
<VALUE>OV_ServiceTypeDefinition.GUID="\DiscoveredElement\"</VALUE>
</PROPERTY.REFERENCE>
<PROPERTY.REFERENCE NAME="Antecedent">
<VALUE>OV_ServiceTypeDefinition.GUID="\DiscoveredElement\"</VALUE>
</PROPERTY.REFERENCE>
<PROPERTY NAME="PropRuleId">
<VALUE>DDK_DefaultPropagationRule</VALUE>
</PROPERTY>
</INSTANCE>
</ManagementModuleElements>
<PolicyElements>
```

```
<PolicyRules>
<INSTANCE CLASSNAME="OV_PolicyRule">
<PROPERTY NAME="Name">
<VALUE>(:If:ExecuteOnUnix#)</VALUE>
</PROPERTY>
<PROPERTY.REFERENCE NAME="Condition">
<VALUE>NULL</VALUE>
</PROPERTY.REFERENCE>
<PROPERTY.REFERENCE NAME="NextCase">
<VALUE>NULL</VALUE>
</PROPERTY.REFERENCE>
</INSTANCE>
<INSTANCE CLASSNAME="OV_PolicyRule">
<PROPERTY NAME="Name">
<VALUE>RunDiscovery@DiscoveredElement</VALUE>
</PROPERTY>
<PROPERTY.REFERENCE NAME="Condition">
<VALUE>OV_PolicyCondition.Name="\WinOS#"</VALUE>
</PROPERTY.REFERENCE>
<PROPERTY.REFERENCE NAME="NextCase">
<VALUE>OV_PolicyRule.Name="(:If:ExecuteOnUnix#)"</VALUE>
</PROPERTY
Y.REFERENCE>
</INSTANCE>
<INSTANCE CLASSNAME="OV_ServiceDiscoveryPolicy">
<PROPERTY.REFERENCE NAME="GroupComponent">
<VALUE>OV_ServiceTypeDefinition.GUID="\DiscoveredElement"</VALUE>
</PROPERTY.REFERENCE>
<PROPERTY.REFERENCE NAME="PartComponent">
<VALUE>OV_PolicyRule.Name="\RunDiscovery@DiscoveredElement"</VALUE>
</PROPERTY.REFERENCE>
</INSTANCE>
</PolicyRules>
```



```
<ActionParts>
<INSTANCE CLASSNAME="OV_ActionPart">
<PROPERTY NAME="Name">
<VALUE>ExecuteOnUnix#</VALUE>
</PROPERTY>
<PROPERTY.REFERENCE NAME="Part">
<VALUE>OV_PartInstance.Name=\ "ExecuteOnUnix\"</VALUE>
</PROPERTY.REFERENCE>
</INSTANCE>
<INSTANCE CLASSNAME="OV_ActionPart">
<PROPERTY NAME="Name">
<VALUE>ExecuteOnWindows#</VALUE>
</PROPERTY>
<PROPERTY.REFERENCE NAME="Part">
<VALUE>OV_PartInstance.Name=\ "ExecuteOnWindows\"</VALUE>
</PROPERTY.REFERENCE>
</INSTANCE>
<INSTANCE CLASSNAME="OV_ContainedActions">
<PROPERTY.REFERENCE NAME="GroupComponent">
<VALUE>OV_PolicyRule.Name=\ (:If:ExecuteOnUnix#)\</VALUE>
</PROPERTY.REFERENCE>
<PROPERTY.REFERENCE NAME="PartComponent">
<VALUE>OV_ActionPart.Name=\ "ExecuteOnUnix#\"</VALUE>
</PROPERTY.REFERENCE>
</INSTANCE>
<INSTANCE CLASSNAME="OV_ContainedActions">
<PROPERTY.REFERENCE NAME="GroupComponent">
<VALUE>OV_PolicyRule.Name=\ "RunDiscovery@DiscoveredElement\"</VALUE>
</PROPERTY.REFERENCE>
<PROPERTY.REFERENCE NAME="PartComponent">
<VALUE>OV_ActionPart.Name=\ "ExecuteOnWindows#\"</VALUE>
</PROPERTY.REFERENCE>
</INSTANCE>
```

```
</ActionParts>
<ConditionParts>
<INSTANCE CLASSNAME="OV_PolicyCondition">
  <PROPERTY NAME="Name">
    <VALUE>WinOS#</VALUE>
  </PROPERTY>
  <PROPERTY NAME="__CLASS">
    <VALUE>OV_ConditionPart</VALUE>
  </PROPERTY>
  <PROPERTY.REFERENCE NAME="Part">
    <VALUE>OV_PartInstance.Name=\ "WinOS\ " </VALUE>
  </PROPERTY.REFERENCE>
</INSTANCE>
</ConditionParts>
<ConditionExpressions />
<PartInstances>
<INSTANCE CLASSNAME="OV_PartInstance">
  <PROPERTY NAME="Name">
    <VALUE>ExecuteOnUnix</VALUE>
  </PROPERTY>
  <PROPERTY NAME="Caption">
    <VALUE>ExecuteOnUnix</VALUE>
  </PROPERTY>
  <PROPERTY NAME="Description">
    <VALUE>ExecuteOnUnix</VALUE>
  </PROPERTY>
  <PROPERTY.REFERENCE NAME="PartTemplate">
    <VALUE>OV_PartTemplate.Name=\ "ActionTemplate\ " </VALUE>
  </PROPERTY.REFERENCE>
  <PROPERTY.REFERENCE NAME="ParameterBlock">
    <VALUE>OV_ParameterBlock.Name=\ "ParameterBlock_ExecuteOnUnix\ " </VALUE>
  </PROPERTY.REFERENCE>
<INSTANCE CLASSNAME="OV_ParameterBlock">
```

```
<PROPERTY NAME="Name">
<VALUE>ParameterBlock_ExecuteOnUnix</VALUE>
</PROPERTY>
<PROPERTY.ARRAY NAME="ConfigPairs">
<VALUE.ARRAY>
<VALUE>CommandLine="/var/opt/OV/bin/instrumentation/vertica_mon_disc.pl"</VALUE>
<VALUE>User=</VALUE>
<VALUE>Password=</VALUE>
<VALUE>Secure=:Password:</VALUE>
</VALUE.ARRAY>
</PROPERTY.ARRAY>
</INSTANCE>
</INSTANCE>
<INSTANCE CLASSNAME="OV_PartInstance">
<PROPERTY NAME="Name">
<VALUE>ExecuteOnWindows</VALUE>
</PROPERTY>
<PROPERTY NAME="Caption">
<VALUE>ExecuteOnWindows</VALUE>
</PROPERTY>
<PROPERTY NAME="Description">
<VALUE>ExecuteOnWindows</VALUE>
</PROPERTY>
<PROPERTY.REFERENCE NAME="PartTemplate">
<VALUE>OV_PartTemplate.Name="ActionTemplate"</VALUE>
</PROPERTY.REFERENCE>
<PROPERTY.REFERENCE NAME="ParameterBlock">
<VALUE>OV_ParameterBlock.Name="ParameterBlock_ExecuteOnWindows"</VALUE>
</PROPERTY.REFERENCE>
<INSTANCE CLASSNAME="OV_ParameterBlock">
<PROPERTY NAME="Name">
<VALUE>ParameterBlock_ExecuteOnWindows</VALUE>
</PROPERTY>
```

```
<PROPERTY.ARRAY NAME="ConfigPairs">
  <VALUE.ARRAY>
    <VALUE>CommandLine="%OvDataDir%/bin/instrumentation/vertica_mon_disc.pl"</VALUE>
    <VALUE>User=</VALUE>
    <VALUE>Password=</VALUE>
    <VALUE>Secure=:Password:</VALUE>
  </VALUE.ARRAY>
</PROPERTY.ARRAY>
</INSTANCE>
</INSTANCE>
<INSTANCE CLASSNAME="OV_PartInstance">
  <PROPERTY NAME="Name">
    <VALUE>WinOS</VALUE>
  </PROPERTY>
  <PROPERTY NAME="Caption">
    <VALUE>WinOS</VALUE>
  </PROPERTY>
  <PROPERTY NAME="Description">
    <VALUE>WinOS</VALUE>
  </PROPERTY>
  <PROPERTY.REFERENCE NAME="PartTemplate">
    <VALUE>OV_PartTemplate.Name="\BasicDiscTemplate"</VALUE>
  </PROPERTY.REFERENCE>
  <PROPERTY.REFERENCE NAME="ParameterBlock">
    <VALUE>OV_ParameterBlock.Name="\ParameterBlock_WinOS"</VALUE>
  </PROPERTY.REFERENCE>
  <INSTANCE CLASSNAME="OV_ParameterBlock">
    <PROPERTY NAME="Name">
      <VALUE>ParameterBlock_WinOS</VALUE>
    </PROPERTY>
    <PROPERTY.ARRAY NAME="ConfigPairs">
      <VALUE.ARRAY>
        <VALUE>DiscoveryType=OSQuery</VALUE>
```

```
<VALUE>Name=Windows</VALUE>
</VALUE.ARRAY>
</PROPERTY.ARRAY>
</INSTANCE>
</INSTANCE>
</PartInstances>
<PartTemplates>
<INSTANCE CLASSNAME="OV_PartTemplate">
<PROPERTY NAME="Name">
<VALUE>ActionTemplate</VALUE>
</PROPERTY>
<PROPERTY NAME="ExecutionModule">
<VALUE>com.hp.openview.OvDiscoveryParts.OvActionPart</VALUE>
</PROPERTY>
<PROPERTY NAME="ParameterTemplate">
<VALUE>&lt;PartParams&gt;&lt;Param name="CommandLine"&gt;&lt;Caption&gt;Command
Line&lt;/Caption&gt;&lt;UserEditable&gt;&lt;UserEdit&gt;&lt;Values&gt;&lt;User
Value&gt;&lt;/Values&gt;&lt;/Param&gt;&lt;Param name="User"&gt;&lt;Caption&gt;U
ser Name&lt;/Caption&gt;&lt;Optional&gt;&lt;UserEditable&gt;&lt;UserEdit&gt;&lt;
Values&gt;&lt;UserValue&gt;&lt;/Values&gt;&lt;/Param&gt;&lt;Param name="Pass
word"&gt;&lt;Caption&gt;Password&lt;/Caption&gt;&lt;Optional&gt;&lt;UserEditabl
e&gt;&lt;UserEdit&gt;&lt;Securable&gt;&lt;Secure&gt;&lt;Values&gt;&lt;UserVa
lue&gt;&lt;/Values&gt;&lt;/Param&gt;&lt;/PartParams&gt;</VALUE>
</PROPERTY>
</INSTANCE>
<INSTANCE CLASSNAME="OV_PartTemplate">
<PROPERTY NAME="Name">
<VALUE>BasicDiscTemplate</VALUE>
</PROPERTY>
<PROPERTY NAME="ExecutionModule">
<VALUE>com.hp.openview.OvDiscoveryParts.OvBasicDiscoveryPart</VALUE>
</PROPERTY>
<PROPERTY NAME="ParameterTemplate">
<VALUE>&lt;PartParams&gt;&lt;Param name="DiscoveryType"&gt;&lt;Caption&gt;Discov
ery Type&lt;/Caption&gt;&lt;Values default="File"&gt;&lt;Value name="File"&gt;&lt;
t;Params&gt;&lt;Param
```

```
name="FileName"><Caption>Filename</Caption><Values><User
Value/></Values></Param></Params></Value><Value na
me="Process"><Params><Param name="ProcessName"><Caption>Pro
cess Name</Caption><Values><UserValue/></Values></Par
am></Params></Value><Value name="Registry"><Params><l
t;Param name="Key"><Caption>Key</Caption><Values><UserVa
lue/></Values></Param><Param name="Value"><Caption>Va
lue</Caption><Optional/><Values><UserValue/></Values>
</Param><Param name="TestValue"><Caption>Test Value</Cap
tion><Optional/><Values><UserValue/></Values></Par
am></Params></Value><Value name="WMIQuery"><Params><l
t;Param name="Namespace"><Caption>Namespace</Caption><Values>
<UserValue default="root\cimV2"/></Values></Param><Param
name="Query"><Caption>Query</Caption><Values><UserValue/
></Values></Param></Params></Value><Value name="OS
Query"><Params><Param name="Architecture"><Caption>Architec
ture</Caption><Optional/><Values><UserValue/></Values>
</Param><Param name="Name"><Caption>Name</Caption><l
t;Optional/><Values><UserValue/></Values></Param><
Param name="Version"><Caption>Version</Caption><Optional/>
<Values><UserValue/></Values></Param></Params></
Value></Values></Param></PartParams></VALUE>

</PROPERTY>

</INSTANCE>

</PartTemplates>

</PolicyElements>

</AutoDiscPolicy>
```

Glossary

B

BSM Operations Management

BSM Operations Management is the event management foundation for a complete BSM monitoring solution. As the operations bridge, it consolidates all IT infrastructure monitoring in a central event console, and relates the events to the IT services that depend on that infrastructure. Users benefit from a common structured event management model that applies the same processes to both business service management and IT infrastructure management.

C

Configuration Item (CI)

As the ITIL definition goes, Any Component that needs to be managed in order to deliver an IT Service. CIs typically include IT Services, hardware, software, buildings, people and formal documentation such as Process documentation and SLAs. Configuration Item Type (CIT) describes the type of a CI and its attributes.

M

Monitoring Automation (MA)

Monitoring automation provides a complete management solution for an application or service, enabling you to create a management solution for the entire set of configuration items (CIs) comprising the application. The solution can be made to respond dynamically to changes in the topology, making the monitoring solution independent of the hardware and platform running the application.

R

Run-time Service Model (RTSM)

Run-time Service Model (RTSM) which is part of BSM contains all CIs and relationships created in BSM. The CIs and relationships together represent a model of all the components of the IT infrastructure.

T

Topology-Based Event Correlation (TBEC)

The event management process is simplified by categorizing events using topology-based event correlation (TBEC). Dependencies between events are analyzed to determine whether some events can be explained by other events. If two events occur concurrently (within a configurable time span), TBEC correlation rules identify one event as the cause and the other event to be the symptom.

We appreciate your feedback!

If you have comments about this document, you can [contact the documentation team](#) by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Development Guide (OMi Management Pack 1.00)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to docfeedback@hp.com.

