Migrating to BMC Event Manager from IBM Tivoli Enterprise Console
Executive Summary
Organizations that want to move away from IBM Tivoli Enterprise Console should consider migrating to BMC Event Manager. BMC Event Manager has numerous advantages over Tivoli Enterprise Console in terms of ease of use, administration costs, and performance, and it provides a natural gateway into service impact management capabilities and service support integrations. More importantly, BMC is an obvious choice because the BMC Event Manager technology naturally fits as a replacement for Tivoli Enterprise Console since it is fully compatible with the IBM Tivoli event sources and uses similar, though easier, paradigms for event management.

This document reviews the similarities that make migration easy, as well as the advantages the BMC solution offers to customers seeking a low-cost, effective alternative as they look to “sunset” their Tivoli Enterprise Console implementation without throwing away existing integrated products, skills, processes, and intellectual property.

Section 1 of this document outlines why the migration process is easy:

> Tivoli Enterprise Console and BMC Event Manager both use BAROC and IBM Tivoli Event Integration Facility (EIF) as event communication standards.
> BMC Event Manager and Tivoli Enterprise Console use similar paradigms in terms of rules.

Section 2 describes additional benefits recognized after migrating to BMC Event Manager:

> Lower administration costs
> Higher performance
> Service support integration
> Service impact management

Section 3 provides an outline of the steps involved in a typical migration project.

Terminology Note
BMC Event Manager is one of two products based on BMC Impact Manager technology. BMC Service Impact Manager is the other. Therefore, BMC Event Manager and BMC Service Impact Manager share the same rule engine and the same interfaces (CLI, UI, API).

In this document, it is often easier to talk about BMC Impact Manager when a feature is available in both the BMC Event Manager and BMC Service Impact Manager products.
1. Why is Migration Easy?
When you travel from one country to another with your laptop, you may face issues around the fact that the electrical systems differ: voltages and frequency vary, and there is also a good chance you left home without the right power-plug adaptor for your destination. It is not very different for event transport protocols: if they differ, you will have to worry about building or buying specific adapters and transformers (among other things) to ensure your event sources can communicate with the event server.

As Tivoli Enterprise Console and BMC Event Manager use the same “voltage” (EIF) and the same “power-plugs” (BAROC), these considerations do not apply, as we will see below. NetCool technologies, on the other hand, will require adapting voltage and power plugs.

BMC Event Manager Understands EIF
EIF is the application programming interface (API) that allows event sources, such as Tivoli NetView or Tivoli ITM, to connect and send events to a Tivoli Enterprise Console. Since its first version, BMC Impact Manager (including both BMC Event Manager and BMC Service Impact Manager) has been fully compatible with EIF.

BMC Impact Manager supports EIF, and therefore, the BMC Event Manager can receive events from any EIF event source without writing any complex rules or adapters. What’s more, EIF is the “standard” event protocol within IBM Tivoli, and as a result, tremendously speeds up and simplifies the migration from Tivoli Enterprise Console to BMC Event Manager.

Listed below are a few examples of how to integrate popular Tivoli event sources to BMC Event Manager. Essentially, the re-configuration work consists of pointing the event source to the host and port number used by BMC Event Manager (instead of the host and port number of the Tivoli Enterprise Console server). The list of products below is not exhaustive, but rather, a representative sampling of how all monitoring products that support the EIF API can be easily integrated to BMC Event Manager through the following similar steps.

IBM TIVOLI NETVIEW
Older versions of IBM Tivoli NetView relied on the “tecad” adapter for Tivoli Enterprise Console event forwarding. See the “IBM Tivoli Adapters” section on page 4 if you are in this situation.

The current IBM Tivoli Netview has a configuration dialog that lets administrators specify the location of the event server. To use BMC Event Manager in place of the Tivoli Enterprise Console, use the same dialog as follows:

> The **IBM Tivoli Enterprise Console Server host name** must contain the hostname of the BMC Event Manager.

> The **Server port number** must contain the port used by the destination BMC Event Manager (by default 1828).

> **Use TME transport mechanism** must be left to “no”.

>
When a monitor is defined in Tivoli Distributed Monitoring, there are two classical ways to configure it to send an event to Tivoli Enterprise Console:

- Specify a destination Tivoli Enterprise Console server in “Send Enterprise Console event” or
- Specify a program that includes a call to the “postemsg” command

Either way, the logic to replace Tivoli Enterprise Console by BMC Event Manager is similar:

- If the monitor definition uses the “Send Enterprise Console Event,” replace the “EventServer” entry in the corresponding text box with the hostname and port number of the destination instance of BMC Event Manager, using the notation: “hostname+port” (e.g. “winsvr1+1828”).
- If an event is sent using the “run program” option and a script based on the “postemsg” command, the “postemsg” command options can be modified to address a destination BMC Event Manager instance. Alternatively the “postemsg” command can be replaced with the richer “msend” command that is the BMC Event Manager functional equivalent of “postemsg.”

IBM TIVOLI MONITORING 6.1
The configuration (location and port) of the destination Tivoli Enterprise Console server is available on the monitoring server(s) of ITM 6.1.

- On Microsoft Windows, use the “Reconfigure” option inside the Tivoli Enterprise Monitoring Services to reconfigure the destination host and port.
- On UNIX® and Linux® systems, use the “itmcmd” utility to perform the (re)-configuration.

IBM TIVOLI ADAPTERS
Adapters are agents that translate information coming from third-party sources, such as SNMP, log files, or others, into Tivoli Enterprise Console events. Pointing an adapter to a BMC Event Manager simply consists in editing the configuration file (the “.conf” file) of the adapter and providing the following values:

```plaintext
ServerName = <hostname of BMC Event Manager server>
ServerPort = <TCP port of the BMC Event Manager server>
```

Of course, this capability does not stop at the communication level. BMC Event Manager also uses the same event format.

Tivoli Enterprise Console and BMC Event Manager Speak the Same Language
Tivoli Enterprise Console and BMC Event Manager events are encoded using the BAROC language. This means that an event formatted for Tivoli Enterprise Console can be directly consumed and processed by BMC Event Manager.

As the meta-class “Tec_Event” is defined out of the box within BMC Event Manager, and as the a superclass “EVENT” is common to Tivoli Enterprise Console and BMC Event Manager, all the BAROC
files that are part of the Tivoli Enterprise Console rule base can be directly imported in the BMC Event Manager Knowledge Base.\(^1\)

**Tivoli Enterprise Console Rules Easily Translate into BMC Event Manager Rules**

You don’t have to throw away your investments and intellectual property on event management rules; just translate them.

BMC Event Manager includes a ProLog rule engine that shares commonalities with the Tivoli Enterprise Console rule engine. However, the BMC Event Manager rules language (Master Rule Language, or MRL) hides the complexity and the syntax difficulties of ProLog — and the common underlying technology of BMC Impact Manager (including both BMC Event Manager and BMC Service Impact Manager) and Tivoli Enterprise Console make rules conversion much more straightforward. Administrators familiar to Tivoli Enterprise Console rules writing will have no trouble figuring out how Tivoli Enterprise Console rules can be translated into BMC Event Manager rules.

In addition, MRL adds a wealth of additional capabilities in terms of rules writing, so a migration from Tivoli Enterprise Console to BMC Impact Manager provides an opportunity to implement more efficient, generic rules. Among other important enhancements, MRL also provides:

- A straightforward way to use dynamic data within rules
- A specialized, streamlined approach that makes it easier to write rules that address frequently encountered scenarios (such as filtering, de-duplication, and escalation)
- A simplified syntax that hides the specifics of ProLog
- A much easier approach to “change” rules and the “redo_analysis” concept
- A wealth of additional operators and functions
- And much more

See the section on “Lower Administration Costs” on page 8, for more details on “using dynamic data.”

---

\(^1\) Multiple inheritance, a rarely used feature used in the IBM Tivoli version of BAROC, is not supported in BMC Impact Manager. There are easy workarounds in case an IBM BAROC definition using multiple inheritance needs to be imported.
In the following paragraphs, you will find some examples of rules translation that provide a flavor of just how intuitive the rule translation from Tivoli Enterprise Console to MRL is. However, please note that MRL code examples throughout the document are provided as an “induction” into what MRL offers and how it is structured. Full documentation on MRL can be found in the “BMC Impact Solutions – Knowledge Base Development” guide (see Appendix 1). For additional examples, refer to Appendix 2.

**A SIMPLE ENRICHMENT RULE**

The following Tivoli Enterprise Console rule sets the severity of an incoming event of type AgentRestarted to “HARMLESS” if the event comes from host testserver.

```plaintext
rule:change_severity:
    event: _event of_class 'AgentRestarted'
    where
        hostname: equals 'testserver',
    reception_action:set_to_harmless:
        bo_set_slotval(_event, severity, 'HARMLESS'),
        re_mark_as_modified(_event)
).```

The same rule written in MRL would be:

```plaintext
refine change_severity:
    AgentRestarted ($EV) where [$EV.hostname == testserver ]
{ $EV.severity = HARMLESS ;
} END```

MRL uses specialized rules to cater to the different event management scenarios. Rules of type “refine” are used whenever an incoming event needs to be enriched. Specialized rules improve performance and make rules development much easier.

“$EV” is an arbitrary variable name that acts as a pointer to the entire event structure. It makes the rule much more straightforward to write — and read.
A DE-DUPICATION RULE
This is a classic “de-duplication rule” as written in the Tivoli Enterprise Console rule language.

```
rule: receive_down_rl:
{
    event: _event of_class 'TWS_DOWN'
    where [ hostname: _hostname,
            date_reception: _date_reception,
            date: _date,
            msg: _msg,
            severity: _severity,
            repeat_count: _repeat_count
    ],
    reception_action:
    {
        first_duplicate(_event, event: _dup_event
            where [ status: outside [ 'CLOSED' ]
                    ])
        add_to_repeat_count(_dup_event, 1),
        bo_set_slotval(_dup_event, msg, _msg),
        bo_set_slotval(_dup_event, severity, _severity),
        bo_set_slotval(_dup_event, date, _date),
        drop_received_event,
        commit_set
    }
}.
```

The MRL equivalent is much simpler to write:

```
new received down rl:
TWS_DOWN ($NEW)
updates duplicate ($OLD) where [$OLD.status != CLOSED]
{
    $OLD.msg = $NEW.msg;
    $OLD.severity = $NEW.severity;
    $OLD.date = $NEW.date
    incr($OLD.repeat_count);
    drop_new;
}
END
```

MRL rules of type “new” (such as the one above) are especially designed for retrieving duplicates or older events. Specialized rule structures enhance the clarity and the performance of the rules.
2. Other Migration Benefits

As seen in the previous section, thanks to the protocol compatibility and similar rules logic, migration of Tivoli Enterprise Console functionalities into BMC Event Manager is a straightforward exercise. However, BMC Event Manager also brings significant advantages in terms of administration costs, scalability, and functionality.

Lower Administration Costs

**Dynamic Rules Make Administration Easier**

Since its first version, BMC Event Manager has included the capability to use dynamic data inside its rules. This feature, named Dynamic Data Association (DDA), allows a clear separation between the event processing logic and the context information. Hostnames, addresses, application names and dependencies, escalation policies, and more are stored in external tables that can be updated dynamically, directly from the BMC Event Manager console (BMC Impact Explorer) or from external sources. The processing rules use these tables to make decisions on the events. This capability tremendously reduces the administration costs traditionally associated to the maintenance of the management rules.

Tivoli Enterprise Console has only recently included similar capabilities, which still require advanced understanding of the Tivoli Enterprise Console rule language and do not provide the same flexibility and ease of use.

The ability of BMC Event Manager to work in a data-driven mode has been taken a step further with the implementation of event management policies. Event management policies let administrators dynamically define event processing rules for classical scenarios, such as blackout, filtering, escalation, and notification. Policies once again significantly reduce the complexity and cost of managing events, allowing administrators to create new polices with just a few clicks inside the BMC Event Manager user interface.

**Unique Architecture Lowers Infrastructure Costs**

In addition to the above features, BMC Event Manager has a very lightweight architecture, which does not require a database to run it. BMC Event Manager CPU and memory usage can also be tuned to adjust to low-end hardware, where excellent performance can already be obtained.

*(NOTE: If events need to be stored in a database, there is a separate BMC Impact Database Gateway product that allows you to store BMC Event Manager events in an external repository.)*

Superior Scalability

Each BMC Event Manager instance is able to process up to hundreds of events per second, which is much higher than typical Tivoli Enterprise Console throughputs. Moreover, multiple instances of BMC Event Manager can be distributed across the environment to match topology and volume constraints of large organizations. Automatic synchronization and failover capabilities make this distributed architecture very robust and much easier to set up and maintain than a distributed Tivoli Enterprise Console infrastructure.
Service Impact Management
BMC Service Impact Manager uses events to measure and prioritize the impact of infrastructure issues on the business services of the organization. It adds business context and impact information to the events and therefore helps operations and service support to prioritize actions based on impact and not just event severity. BMC Service Impact Manager also assists in identifying the probable cause of a service disruption.

Figure 1 shows a typical BMC Event Manager screen. Events are color-coded by severity (as in Tivoli Enterprise Console), but the severity does not indicate what the business impact is.

BMC Service Impact Manager applies the events received by BMC Event Manager to a service model that describes the dependencies among the services and the infrastructure that generates these events. Based on the incoming events and the model, BMC Service Impact Manager is then able to determine the business impact of the technical issue.

Figure 2 (on page 10) shows a typical BMC Service Impact Manager screen — available as another tab of the BMC Impact Explorer (which also includes the BMC Event Management console shown in Figure 1). Only one of the events shown above has an impact on the “South America Sales” defined in BMC Service Impact Manager.
BMC Service Impact Manager uses the same “BAROC” events as BMC Event Manager and Tivoli Enterprise Console, so the effort to add service impact management capabilities will be minimal — with huge benefits in terms of operational efficiency and service quality.

Service Support Integration
BMC Impact Manager (both BMC Event Manager and BMC Service Impact Manager) tightly integrates with BMC Remedy Service Desk for intelligent incident creation — either based on events (when BMC Event Manager is used) or service impact information (when BMC Service Impact Manager is used). This integration works out of the box and does not require the creation of complex custom workflow and rules that are required to integrate with Tivoli Enterprise Console.

Using the Integration to BMC Remedy Service Desk, incidents can be created out-of-the-box:

- Manually (for individual events; simply “right-click” on selected events)
  - See Figure 3, page 11
- Automatically (for events matching some conditions, such as “all critical events”)
- Automatically or manually (when a service is impacted)
  - This is in case BMC Service Impact Manager is used as well
MIGRATING TO BMC EVENT MANAGER FROM IBM TIVOLI ENTERPRISE CONSOLE

3. Sample Migration Strategies

This section provides an outline of the main steps involved in a Tivoli Enterprise Console to BMC Event Manager migration. First, it lists the main information elements that must be gathered prior to starting the migration process. Then, it provides additional information on the migration tasks themselves. Finally, it presents two possible strategies, including the advantages and drawbacks of each.

(Please note: This is not a replacement for a proper consulting services engagement. Project management and planning aspects, such as phasing, hardware acquisition, training, and others, are not included here.)

Gathering Information

Regardless of the migration strategy, the following audit and analysis of the existing environments must be performed.

**REVIEW THE GLOBAL TIVOLI ENTERPRISE CONSOLE ARCHITECTURE**

A clear, detailed view on the Tivoli Enterprise Console architecture must be obtained. This should include:

> An inventory and diagram of all Tivoli Enterprise Console servers and how they are connected
> The location of all the clients (event sources, consoles, and other integrations)
> A description of the backup and failover procedures in place
INVENTORY ALL EVENT SOURCES
All Tivoli Enterprise Console event sources must be identified. The inventory should include:

> **The nature of each event source**
  Is the event provider a Tivoli product? Third party? Customization? Other?

> **The location of the event source instances**
  Are there multiple instances of a same event source? Where are they located?

> **The format of the events and their volumes**
  How many events are generated daily? What is the BAROC specification for these events?

> **The Tivoli Enterprise Console rules that specifically cater to these events**
  What specific processing rules apply to these events (ie. enrichment, normalization, de-duplication, etc.)?

REVIEW THE CURRENT TIVOLI ENTERPRISE CONSOLE RULE BASE
The Tivoli Enterprise Console rule base essentially consists of:

> **BAROC class definitions**
  As seen previously, these definitions can normally be re-used “as is” in BMC Event Manager.

> **Rules**
  All rules must be reviewed for translation purposes. Special attention must be provided to rules that embed native ProLog code.

INVENTORY ALL DOWNSTREAM INTEGRATIONS
By “downstream” integrations, we mean integrations to applications that consume the Tivoli Enterprise Console events, such as notification systems, service support applications, reporting applications, etc. Generally these integrations make use of one of the following mechanisms:

> **Event forwarding**
  Events are sent downstream through the Tivoli Enterprise Console “forward_event()” statement.

> **Script execution**
  A custom script (shell, DOS, etc) is used as a wrapper around an executable that connects to the other application.

> **Other**
  In some (less frequent) situations, the integration is made e.g. at Tivoli Enterprise Console database level (by querying the Tivoli Enterprise Console event tables) or using other mechanisms.

ANALYZE CONSOLE REQUIREMENTS.
The Tivoli Enterprise Console event console is the primary user interface for operators who deal with Tivoli Enterprise Console events. With migration in mind, the following information must be gathered:

> **Who are the users of the Tivoli Enterprise Console event console?**
  How many operators are there? Where are they located?

> **What are the event visualization requirements?**
  How are events presented and organized in each operator’s console?
What are the event interaction requirements?
Do operators manually acknowledge/close events, invoke programs, tasks, etc.?

Migration
This document has already provided an outline on how to “point” classical Tivoli event sources to BMC Event Manager, as well as examples of rules translation. This section contains additional considerations on the main functional blocks that need be migrated.

DESIGN A BMC EVENT MANAGER ARCHITECTURE
At this point, you may be asking yourself how many BMC Event Manager instances should be deployed and where they should be located. The simple ground rule is to replace each Tivoli Enterprise Console server by a BMC Event Manager instance. However, BMC Event Manager brings three specific architectural advantages over Tivoli Enterprise Console that may alter this ground rule:

1) BMC Event Manager usually has much better event throughput than the Tivoli Enterprise Console. Therefore, if multiple, parallel Tivoli Enterprise Console servers were deployed for the sole purpose of coping with high event volumes in a single location, this may no longer be required, as a single BMC Event Manager instance could consume the load of several Tivoli Enterprise Console servers.

2) Multiple autonomous BMC Event Manager instances can be easily distributed across the infrastructure, as they are able to natively communicate and synchronize with each other. These features are superior to the capabilities of Tivoli Enterprise Console in that area, hence a much easier approach to a distributed event management solution.

3) BMC Impact Manager (i.e. BMC Event Manager and BMC Service Impact Manager) includes native failover capabilities.2 Implementing a high-availability solution is therefore much easier than with Tivoli Enterprise Console, where this capability requires investment in clustering technologies.

In larger, distributed environments, it is worth considering the use of additional BMC Event Manager instances to process events closer to the event sources [the ground rule would be: one BMC Event Manager instance per data center] and to provide for a more robust, highly available solution.

The “Architecture and Performance Guidelines” document listed in Appendix 1 provides additional details on reference BMC Event Manager architectures.

---

2 Feature available with BMC Impact Manager (BMC Event Manager, BMC Service Impact Manager) 7.1
BMC Event Manager: Best Practices for Architecture Design

> As a base rule, consider using one BMC Event Manager in place of each Tivoli Enterprise Console server.

> Leverage BMC Event Manager excellent throughput to discard additional Tivoli Enterprise Console servers that are used to cope with high event volumes.

> Leverage BMC Event Manager failover capabilities to build a highly-available solution.

> Refer to the “Architecture and Performance Guidelines” document to design and size your future BMC Event Manager environment.

TRANSLATE THE TIVOLI ENTERPRISE CONSOLE RULE BASE

As already mentioned, the BAROC class definitions of the Tivoli Enterprise Console rule base can be imported “as is” to the target BMC Event Manager knowledge base. The vast majority of Tivoli Enterprise Console rules fall, in one way or another, into the examples provided in the document. However, there is one particular situation where translation might be more challenging: whenever custom ProLog is used inside Tivoli Enterprise Console rules, further analysis on how best to translate this logic is needed.

Other best practices follow:

BMC Event Manager: Best Practices for Rules Translation:

> Take the opportunity for some housekeeping: discard old, useless rules, or regroup rules that have similar purposes.

> Consider the usage of the graphical “Event Management Policies” for simple event automation tasks, such as filtering, notification, etc.

> Use the BMC Event Manager Dynamic Data Association feature to remove all hard-coded references in the rules.

> Use a similar grouping of rules within rule sets when translating rules.

> Use the examples in this document for the translation of frequently encountered rules.

> Note that MRL does not support native ProLog code, so alternatives must be researched when ProLog is used within Tivoli Enterprise Console rules.
INTEGRATE BMC EVENT MANAGER WITH DOWNSTREAM APPLICATIONS
As previously described, there are only a limited number of ways to integrate Tivoli Enterprise Console with "downstream" application:

> **Event forwarding**
   Event forwarding is natively supported inside BMC Impact Manager, using propagation rules or the send_to() MRL function.

> **Script execution**
   As seen in the “Automated execution” rules translation example in Appendix 2, MRL provides an equivalent to the “exec_program” function of the Tivoli Enterprise Console rule language.

Other integration mechanisms need to be analyzed in detail for migration in the BMC Impact Manager space. For example, if database-level integration is required, BMC Event Manager can export its events to a relational database using an additional module named “BMC Impact Database Gateway.”

Note also that BMC Event Manager includes its own reporting capability (named BMC Impact Reporting) for canned event reports.

PORT CONSOLE REQUIREMENTS INTO BMC IMPACT EXPLORER
BMC Impact Explorer is the main operational console into BMC Impact Manager (including both BMC Event Manager and BMC Service Impact Manager), and it offers a feature-rich user interface:

> BMC Impact Explorer can be installed as a stand-alone application or as a Web-based application.
> BMC Impact Explorer includes a fine-grained security model to restrict user-level access to console and events.
> BMC Impact Explorer includes a rich set of event operations (acknowledge/close, script execution, UI cross launch, notes addition, online help linkage, etc.).
> Events can be dynamically classified using arbitrary factors (such as hostname, etc).
> Event views provide a graphical representation of events.
> The same BMC Impact Explorer user interface is common to BMC Event Manager and BMC Service Impact Manager.

This set of features usually supersedes the functionalities available inside the Tivoli Enterprise Console console. The screenshots on the following pages give a flavor of the various views that BMC Impact Explorer offers.
MIGRATING TO BMC EVENT MANAGER FROM IBM TIVOLI ENTERPRISE CONSOLE

Figure 4. Classic BMC Impact Explorer event console

Figure 5. Event graphical views inside BMC Impact Explorer
Redirect Event Sources from Tivoli Enterprise Console to BMC Event Manager

Examples of how to reconfigure classical event sources from Tivoli Enterprise Console to BMC Event Manager have been provided in previous sections. Other event sources follow the same basic principles.

Migration Strategies

Here are two examples of how migration can be approached. Once again, the outline provided here is no replacement for a proper consulting services engagement. If the target BMC Event Manager architecture (in terms of topology and number of BMC Event Manager instances) significantly differs from the starting Tivoli Enterprise Console architecture, additional considerations apply.

Option 1 – Parallel Environments

In this first approach, a parallel BMC Event Manager environment is built "on the side" of the current Tivoli Enterprise Console environment, which remains fully in production. The migration of the “server” part of the solution (rules, integrations, consoles) is performed in parallel to ongoing operations.

Production event sources that support event forwarding to multiple parallel destinations are reconfigured to also forward events to a test BMC Event Manager instance (in addition to the production destination Tivoli Enterprise Console server). For event sources that do not support event forwarding to multiple destinations, a test environment must be built.
When the entire BMC Event Manager solution is ready, it is brought into the production environment. Production event sources are then all directed to the destination BMC Event Manager(s).

**ADVANTAGES**
> Only one environment (Tivoli Enterprise Console, then BMC Event Manager) is in production at any time.
> The existing Tivoli Enterprise Console environment is not modified until cutoff; any issue in the migration exercise does not disrupt operations.

**DRAWBACKS**
> Additional hardware is required for the parallel environment.
> The cutoff between environments is immediate and the learning curve for operators may be steep when changing consoles.
> As the cutoff is immediate, there are some risks associated to the moment of the transition.

**OPTION 2 – STAGED MIGRATION**
This second approach leverages the existing BMC Impact Integration for Tivoli integration module that provides for event forwarding from Tivoli Enterprise Console to BMC Event Manager.

Using this option, the production Tivoli Enterprise Console server forwards all its events to a BMC Event Manager instance that is used in “pre-production” mode. Event sources, as well as rules and integrations, are migrated one by one from Tivoli Enterprise Console to BMC Event Manager, until Tivoli Enterprise Console does not receive any more events and can be retired.

**ADVANTAGES**
> This strategy provides an incremental, more phased approach to migration.
> Operators can transition more easily from the Tivoli Enterprise Console to the BMC Event Manager console. The learning curve is flatter.
> There are fewer risks associated to the migration due to its phased nature.

**DRAWBACKS**
> This strategy requires installing and implementing BMC Impact Integration for Tivoli.
> Two distinct production environments and consoles co-exist.
4. Conclusion

Many IT operations organizations utilizing IBM Tivoli Enterprise Console for event management are finding that they no longer receive the desired performance required to support their environment and the demands of the business for high service levels. They are seeking a more flexible approach to managing their operational infrastructure, and BMC Event Manager provides a powerful, more flexible, cost-effective approach.

In addition, organizations with Tivoli Enterprise Console that are implementing Business Service Management (BSM) initiatives will highly benefit from migrating to BMC Event Manager. BMC Event Manager is the springboard into service impact management and provides tighter integration with the service support organization.

With the common protocols and paradigms of Tivoli Enterprise Console and BMC Event Manager, migration between these two technologies is relatively straightforward.

For more information on BMC Event Manager, please visit www.bmc.com/eventmanager.
Appendix 1: Additional Resources

Master Rule Language (MRL) code examples throughout the document are provided as an “induction” into what MRL offers and how it is structured. Full documentation on MRL can be found in the “BMC Impact Solutions – Knowledge Base Development” guide:


Reference architectures for BMC Impact Solutions can be found under the “BMC Impact Solutions 7.0.02 – Architecture and Performance guidelines document” under:


Appendix 2: More Rules Translations Examples

As a complement to the examples given in the body of the document, here are additional examples of how Tivoli Enterprise Console rules would translate into MRL rules.

An Enrichment Rule

The following rule applies to incoming events of class “WebSphere MQ_ChannelError”. The rule reformats the event message using strings contained in different slots of the event.

```plaintext
rule:
  mq_formatmsg:
  {  
    event: _event of_class 'WebSphere_MQ_ChannelError',
    where [  
      channel_name: _channel_name,
      profilename: _profilename,
      status: outside ["CLOSED"],
    ],
    reception_action: (  
      set_event_message(_event, '%s = %s', [_profilename, _channel_name]),
      re_mark_as_modified(_event,)
    ),
  }.
```

The same rule written in MRL would be:

```plaintext
refine mq_formatmsg:
  WebSphere_MQ_ChannelError ($EV)
  {
    $EV.msg = $EV.profilename || " = " || $EV.channel_name;
  }
END
```

In MRL, there is no need to specify that “the value has changed” (re_mark_as_modified). Any change will be automatically detected by other rules if needed.

A Timer Rule

The concept of timers is very similar in BMC Event Manager and Tivoli Enterprise Console. In this example, the first rule sets a timer of 60 seconds on events of type TWS_UP. The second rule fires after the expiration of the timer and sets the status of the event to CLOSED.

```plaintext
rule:
  timer:
  {  
    event: _event of_class 'TWS_UP',
    when: system event_time 60,  
    reception_action: (  
      set_event_time(_event, _),
      set_event_status(_event, "CLOSED")
    ),
  }.
```

The same rule written in MRL would be:

```plaintext
refine timer:
  TWS_UP ($EV)
  {
    $EV.time = $EV.time + 60;
    $EV.status = "CLOSED";
  }
END
```
MIGRATING TO BMC EVENT MANAGER
FROM IBM TIVOLI ENTERPRISE CONSOLE

rule: set_timer_on_tws_up:
  (
  event: _event of_class 'TWS_UP'
  reception_action:
    (
      set_timer(_event, 60, "close")
    ),
  ).

timer_rule: close_tws_up_after_timer:
  (  
    event: _event of_class _class
    where [
      status: outside ['CLOSED']
    ],
    timer_info: equals 'close',
    action:
      (  
        change_event_status(_event, 'CLOSED'),
        commit_set
      )
  ).

The translation is immediate in MRL:

new set_timer_on_tws_up:
TWS_UP ($EV)
triggers
{ set_timer ($EV, 60, "close"); }
END

timer close_tws_up_after_timer:
EVENT ($EV) where [$EV.status != CLOSED]
timer_info: equals close
{  
  $EV.status = CLOSED;
}
END

Automatic Execution
The automation of program execution is an important aspect of event management. Once again, MRL offers a straightforward way of translating Tivoli Enterprise Console rules that call external executables.

The following rule calls a specific script upon reception of events of type “Sentry2_0_scustm”. The script itself uses three slots (hostname, probe_arg and value) as positional arguments.
The MRL translation would be similar – just simpler:

```mrl
new qa_run_modem:
Sentry2_0_scustom ($EV)
triggers
{
  execute($EV, "QARUN.sh", [$EV.hostname,$EV.probe_arg,$EV.value], NO);
}
END
```

A Change Rule
MRL natively detects changes and there is no need to call specific “change” rules to trigger further processing when a slot changes. Rather, the functionality is directly available trough the “when” clauses, which can be positioned at any place in the code.

The first Tivoli Enterprise Console rule below fires when a “MAINTENANCE_START” event is received and calls the execution of the program “maintenance_start.sh”.

The second Tivoli Enterprise Console rule fires when the same event is set to CLOSED and calls another executable.

```mrl
rule: execute_script_on_receive:
{
  event: _event of_class ‘MAINTENANCE_START’,
  reception_action:
  {
    exec_program(_event,'/usr/Tivoli/TEC/scripts/maintenance_start.sh','',[],'NO')
  }
}
```
In MRL, the code is more straightforward and requires only one rule:

```rml
execute execute_for_maintenance:
MAINTENANCE_START ($EV)
when $EV.status == OPEN
{
  execute($EV,"maintenance_start.sh",[],NO);
}
when $EV.status == CLOSED
{
  execute($EV,"maintenance_stop.sh",[],NO);
}
END
```
BMC Software delivers the solutions IT needs to increase business value through better management of technology and IT processes. Our industry-leading Business Service Management solutions help you reduce cost, lower risk of business disruption, and benefit from an IT infrastructure built to support business growth and flexibility. Only BMC provides best-practice IT processes, automated technology management, and award-winning BMC Atrium technologies that offer a shared view into how IT services support business priorities. Known for enterprise solutions that span mainframe, distributed systems, and end-user devices, BMC also delivers solutions that address the unique challenges of the midsized business. Founded in 1980, BMC has offices worldwide and fiscal 2007 revenues of $1.58 billion. Activate your business with the power of IT. www.bmc.com.