

*Service Management Delivery Services*

**BSS Integration**

**Cell Performance Policy**

**Low Level Design Documentation (LLD)**

**Prepared for:**

****

**V1.0**

**Final**

© Innovise ESM 2011Document Control

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Version  Number | Revision  Date | Summary of Changes  (List the reason for each version of the document) | Author(s) |
| V1.0 |  | Original |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction 4](#_Toc309995680)

[1.1 Document Objective 4](#_Toc309995681)

[2 High Level Algorithm 5](#_Toc309995682)

[2.1 Describe function 5](#_Toc309995683)

[3 Pre-requisites 6](#_Toc309995684)

[3.1 Rules File 6](#_Toc309995685)

[3.2 TSRM Classification 6](#_Toc309995686)

[3.3 Data Sources 6](#_Toc309995687)

[3.4 Customized Functions 6](#_Toc309995688)

[4 Policy Details 7](#_Toc309995689)

[4.1 Event Reader 7](#_Toc309995690)

[4.2 Synthetic Event 7](#_Toc309995691)

[4.3 Low Level Algorithm 8](#_Toc309995692)

[4.3.1 Processing A “Cell Down” Problem Event 8](#_Toc309995693)

[4.3.2 Processing A “Cleared Site/Cell Down” Problem Event 9](#_Toc309995694)

[4.4 Low Level Flow Chart 10](#_Toc309995695)

[4.4.1 ML\_ZTE\_PartialSiteDown 10](#_Toc309995696)

[4.4.2 ML\_Cell\_Performance Policy Code 11](#_Toc309995697)

# Introduction

## Document Objective

This LLD intends to describe the requirements for BSS Cell Performance Policy

# High Level Algorithm

## Describe function

* The following are events that indicate cell related performance issues.
* These events should be suppressed from TT generation for a specified period of time, to be held in an external table defining the time to suppress for each site type.
* If the event clears within this period then no further action is required.
* If the event is still outstanding after the specified time period has elapsed, a TT should be generated

# Pre-requisites

## Rules File

The EventId is based on the event Summary, as shown in the table below.

|  |  |
| --- | --- |
| **EventId** | **Summary** |
| CellPer\_001 | Cell interruption alarm |

## TSRM Classification

The following classification should be added in TSRM

|  |  |
| --- | --- |
| **EventId** | **Classification** |
| CellPer\_001 | ??????? |

## Data Sources

The following Netcool/Impact data types are required for this policy:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Impact Data Source** | **Impact Data Type** | **Data Base Name** | **Table Name** | **Fields** |
| defaultobjectserver | OS\_Status | Object Server | Alerts.status | **Not required** |
|  |  |  |  |  |

## Customized Functions

This policy used 2 customized functions:

* MobiLib.getSleepTime ()
* MobiLib.AddGenericJournal ()

# Policy Details

## Event Reader

**EventReader Name:** ml\_defaulteventreader

**Policy name:** ML\_CellPerformance

**Filter:** ImpactFlag = 4 and EventId = 'CellPer\_001' and HibernateFlag = 0 and Agent != 'Netcool Impact' and MaintFlag in (1,2)

## Synthetic Event

The synthetic event to be raised should be populated with data as shown in the table below

|  |  |
| --- | --- |
| **Field Name** | **Value** |
| Node | @SiteCode |
| Summary | “ZTE Partial Site Down :: x Of The y Cells :: <Node\_List>” or “ZTE Site Down” |
| AlertGroup | 'Cell Performance' |
| AlertKey | @BSC\_Name |
| Severity | 4 |
| Type | 1 |
| FirstOccurrence | Current Time |
| LastOccurrence | CurrentTime |
| Class | 200026 |
| Domain | @Domain |
| ManCity | @ ManCity |
| CovCity | @ CovCity |
| BSC\_Name | @ BSC\_Name |
| Region | @ Region |
| Site | @Site |
| Network | @Network |
| LogTicket | 1 |
| ImpactFlag | 2 |
| OutsourceContractor | @ OutsourceContractor |
| BusImportance | @ BusImportance |
| OmcEms | @ OmcEms |
| MaintFlag | @MaintFlag |
| AdvCorrServerSerial | @ServerSerial |
| EventId | CellPer\_001 |
| OwnerUID | 65534 |
| Agent | Netcool Impact |
| TTHibernate | 60 |
| SiteCode | @SiteCode |
|  |  |
|  |  |
|  |  |
|  |  |

## Low Level Algorithm

### Processing A Cell Performance Event

* Event Enters Policy as defined by Event Reader
* Set @ImpactFlag = 5
* Look for associated Synthetic Event
* If there is an associated Synthetic Event step unto 2nd Hibernation
* (1st) Hibernate 60 seconds
* Check that the Network Event still exists
* If the Network Event does not exist then Exit
* Look for associated Synthetic Event
* If there is an associated Synthetic Event step unto 2nd Hibernation
* Create Synthetic Event
* (2nd) Hibernate 5 seconds
* Get the Synthetic Event details
* Update Network Event with Synthetic Event Details
* Get the number of Network Events associated with this Synthetic Event
* Update Synthetic Event with the number of associated Network Events
* Set @ImpactFlag = 6

## Low Level Flow Chart

### ML\_Cell\_Performance



### ML\_Cell\_Performance Policy Code

/\* Cell Performance Policy

Author: Chris Janes, Innovise Sept 2010

Description:

V 1.0 20100909 Chris Janes Original

V 1.1 20100929 Chris Janes Correlation by BSC added

V 1.2 20101006 Chris Janes NePriority Field replace with BusImportance

EventReader Filter:

EventID = 'CellPer\_001'

\*/

CurrentSerial=@Serial;

Log("CurrentSerial = " + CurrentSerial + " start ");

@ImpactFlag = 5;

ReturnEvent(EventContainer);

paramSummary=@Summary;

paramType=@Type;

paramNodeAlias=@NodeAlias;

paramLastOcc=@LastOccurrence;

paramSerial=@Serial;

if(@Type==2)

{

paramSerial=@ProblemSerial;

}

// Is there an existing Synthetic Event

SyntheticEventDataType = "OS\_Status";

SyntheticEventFilter = "BSC\_Name = '"+ @BSC\_Name + "' and Class = 200026 And Severity > 0";

log("CurrentSerial = " + CurrentSerial + "SyntheticEventFilter = " + SyntheticEventFilter );

SyntheticEvent = GetByFilter(SyntheticEventDataType, SyntheticEventFilter,False);

NumSyntheticEventFilter = Length(SyntheticEvent);

//if so set LogTicket = 1

log("NumSyntheticEventFilter = " + NumSyntheticEventFilter );

if (NumSyntheticEventFilter == 0)

{

Log("No Synthetic Event 1");

site\_name = @Site;

SleepTime = 60;

alarm\_id = 'Cell Performance';

MobiLib.getSleepTime(site\_name, alarm\_id, SleepTime);

// for testing

//SleepTime = 30;

if (@Type == 2)

{

SleepTime = 0;

}

// 20100927 End of Change

@Grade = SleepTime;

ReturnEvent(EventContainer);

log (" TT has not created yet: Hibernate for: " + SleepTime + " seconds ");

// Updating HibernateFlag and return event. HibernateFlag=1 (Hibernated)

@HibernateFlag = 1;

ReturnEvent(EventContainer);

Action\_key = 'CellPerformance' + @ServerSerial + @ServerName + "#" + getdate();

Hibernate(Action\_key, "Sleeping", SleepTime);

//Updating HibernateFlag and return event. HibernateFlag=2 (Awake)

RemoveHibernation(Action\_key);

@HibernateFlag = 2;

ReturnEvent(EventContainer);

// Is the event still there

ExistingCellPerDataType = "OS\_Status";

ExistingCellPerFilter = "ServerSerial = "+ @ServerSerial + " and ServerName = '" + @ServerName + "' And Severity > 0";

log("ExistingCellPerFilter = " + ExistingCellPerFilter );

ExistingCellPer = GetByFilter(ExistingCellPerDataType, ExistingCellPerFilter,False);

NumExistingCellPer = Length(ExistingCellPer);

//if so set LogTicket = 1

log("NumExistingCellPer = " + NumExistingCellPer );

if (NumExistingCellPer == 0)

{

// Event no longer Exists

Log(" Event no longer Exists");

@ImpactFlag = 6;

ReturnEvent(EventContainer);

Exit();

}

else

{

// Is there an existing Synthetic Event?

Log(" Is there an existing Synthetic Event");

SyntheticEventDataType = "OS\_Status";

SyntheticEventFilter = "BSC\_Name = '"+ @BSC\_Name + "' and Class = 200026 And Severity > 0";

log("SyntheticEventFilter = " + SyntheticEventFilter );

SyntheticEvent = GetByFilter(SyntheticEventDataType, SyntheticEventFilter,False);

NumSyntheticEvent = Length(SyntheticEvent);

//if so set LogTicket = 1

log("NumSyntheticEvent = " + NumSyntheticEvent );

if (NumSyntheticEvent == 0)

{

// Create Synthetic Event

Log("Create New Event");

NewEvent = NewEvent(DefaultEventReader);

NewEvent.EventReaderName = "DefaultEventReader";

NewEvent.EventId = 'CellPer\_001';

NewEvent.ImpactFlag = 2;

NewEvent.Domain=EventContainer.Domain;

NewEvent.Site=EventContainer.Site;

// 20101006 Change by Chris Janes of Innovise

// NewEvent.NePriority=EventContainer.NePriority;

NewEvent.BusImportance = EventContainer.BusImportance ;

// 20101006 End of Change

NewEvent.OutsourceContractor=EventContainer.OutsourceContractor;

NewEvent.OmcEms=EventContainer.OmcEms;

NewEvent.Region=EventContainer.Region;

NewEvent.ManCity=EventContainer.ManCity;

NewEvent.CovCity=EventContainer.CovCity;

NewEvent.Agent = "Netcool Impact";

NewEvent.AlertGroup = 'Cell Performance';

NewEvent.AlertKey = EventContainer.BSC\_Name ;

NewEvent.BSC\_Name = EventContainer.BSC\_Name ;

NewEvent.EventType = EventContainer.EventType ;

NewEvent.Network= EventContainer.Network;

NewEvent.Identifier= "CellPer" + @EventType + @BSC\_Name + @Node + @LastOccurrence;

NewEvent.Summary= "Cell Performance Alarm Handling for " + @BSC\_Name + " ";

NewEvent.Severity= 4;

NewEvent.Type= 1;

NewEvent.Tally= 1;

NewEvent.HibernateFlag = 99;

NewEvent.Node= EventContainer.Node;

NewEvent.NodeAlias= EventContainer.NodeAlias;

NewEvent.SyntheticServerName = "";

NewEvent.SyntheticServerSerial = 0;

NewEvent.FirstOccurrence = NowTime ;

NewEvent.LastOccurrence = NowTime ;

NewEvent.LogTicket = 1;

NewEvent.Class = 200026;

AddDataItem('OS\_Status', NewEvent);

Log("New Event Created ");

}

}

}

//while(NumSyntheticEvent == 0)

//{

//Wait % seconds

Hibernate(Action\_key, "Sleeping", 5 );

RemoveHibernation(Action\_key);

//Does the Synthetic event exist?

DataType = "OS\_Status"; // this needs to be setup

Filter = "Class = 200026 and BSC\_Name = '" + @BSC\_Name + "'";

log("Filter1 = " + Filter);

CountOnly = False;

SyntheticEvent = GetByFilter(DataType, Filter, CountOnly);

NumSyntheticEvent = Length(SyntheticEvent );

//}

log("SyntheticEvent [0].ServerName = " + SyntheticEvent [0].ServerName);

log("SyntheticEvent [0].ServerSerial = " + SyntheticEvent [0].ServerSerial);

//Update Source Event

EventContainer.SyntheticServerName = SyntheticEvent [0].ServerName;

EventContainer.SyntheticServerSerial = SyntheticEvent [0].ServerSerial;

ReturnEvent(EventContainer);

// Get Count of cells Down

DataType = "OS\_Status";

Filter = "Class <> 200026 and EventId = 'CellPer\_001' and BSC\_Name = '" + @BSC\_Name + "' And Severity > 0";

log("Filter = " + Filter );

ProblemCells= GetByFilter(DataType , Filter ,False);

NumProblemCells = Length(ProblemCells);

log("NumProblemCells = " + NumProblemCells );

DataType = "OS\_Status";

Filter = "Class = 200026 and EventId = 'CellPer\_001' and BSC\_Name = '" + @BSC\_Name + "' And Severity > 0";

Log("Filter = " + Filter );

UpdateExpression = "Summary = 'Cell Performance Alarm Handling for " + @BSC\_Name + " : " + NumProblemCells + " Cells Down'";

Log("UpdateExpression = " + UpdateExpression );

BatchUpdate(DataType, Filter, UpdateExpression);

Log(" Update Events");

//Update Synthetic Event's Journal

// 20101001 Changed by Chris Janes of Innovise at ML request

// EventTime = localtime(@FirstOccurrence , "dd/MM/yy HH:mm:ss");

EventTime = localtime(@LastOccurrence , "dd/MM/yy HH:mm:ss");

// 20101001 End of Change

se=SyntheticEvent[0].ServerSerial;

MobiLib.AddGenericJournal(se, paramSummary,paramSerial,paramLastOcc,paramNodeAlias,paramType);

log("Return ImpactFlag = 6 to indicate policy has completed");

// This have to be changed in order to be in sync with other policies or other control flag might be added

@ImpactFlag = 6;

ReturnEvent(EventContainer);