**SITE DOWN POLICY**

**Author:** Alex Silva, Innovise ESM AUG 2010

**Description:**

This policy is for Huawei and Motorola site down events: AlertGroup = “LAPD OML Fault” and AlertGroup = “Last RSL Link Failure”.

**EventReader Filter:**

((AlertGroup = "Last RSL Link Failure" OR AlertGroup = "LAPD OML Fault") AND MaintFlag = 1 AND Severity > 3 AND Type = 1 AND ImpactFlag = to be defined )

OR

((AlertGroup = "Last RSL Link Failure" OR AlertGroup = "LAPD OML Fault") and Severity = 0)

**Site Down Synthetic Event details:**

|  |  |
| --- | --- |
| **Field Name** | **Value** |
| Node | Substring(@Node, 0, 7) |
| Summary | Site Down |
| AlertGroup | @AlertGroup |
| Manager | @Manager |
| Severity | 5 |
| Type | 1 |
| LogTicket | 1 |
| Agent | Netcool Impact |
| FirstOccurrence/ LastOccurrence | Getdate() |
| Domain | @Domain |
| Region | @Region |
| ManCity | @ManCity |
| Identifier | Node + Summary + Type |

**Data Type**:

OS\_Status

Site\_id

SleepTime

**Library function**: MobiLib

**Algorithm:**

0 - Is it a Clear Event? If Yes: Clear the synthetic event and EXIT. If No: Continue

1 - Enrich with site priority and sleep time from CCMDB

2 - Is there an existing TT

2.1 - If yes: Update the TT by inserting the current event's details in the journal and EXIT

3 - If No: Sleep using specific sleep time.

3.1 - Has the event cleared?

3.2 - If yes: Exit

4 - If No: Has a synthetic been created?

4.1 - If Yes: Update the TT by inserting the current event's details in the journal and EXIT

5 - If No: Create synthetic event

5.1 - Add details of all associated environmental events and clear them.

**Assumptions:**

StartDayTime = 10:00

EndDayTime = 18:00

**Environmental Event’s** EventType =’environmentalAlarm’ or ‘Environmental Alarm

**Function used from MobiLib:** SiteDownInsertJournal



SITE DOWN POLICY CODE:

/\*    SITE DOWN Policy  
  
    Author: Alex Silva, Innovise AUG 2010  
    Description:  
    This policy is for Huawei and Motorola site down events:  AlertGroup = "LAPD OML Fault" and AlertGroup = "Last RSL Link Failure".  
  
  
    EventReader Filter:  
        ((AlertGroup = "Last RSL Link Failure" OR AlertGroup = "LAPD OML Fault")  AND MaintFlag = 1 AND Severity > 3 AND Type = 1 AND ImpactFlag =  )  
        OR  
        ((AlertGroup = "Last RSL Link Failure" OR AlertGroup = "LAPD OML Fault") and Severity = 0)   
  
    Algorithm:  
        0 - Is it a Clear Event? If Yes: Clear the synthetic events and EXIT. If No: Continue  
        1 - Enrich with site priority and sleep time from CCMDB  
        2 - Is there an existing TT  
            2.1 - If yes: Update the TT by inserting the current event's details in the journal and EXIT  
        3 - If No: Sleep using specific sleep time.  
            3.1 - Has the event cleared?  
            3.2 - If yes: Exit  
        4 - If No: Has a synthetic been created?  
            4.1 - If Yes: Update the TT by inserting the current event's details in the journal and EXIT  
        5 - If No: Create synthetic event  
            5.1 - Add details of all associated environmental events and clear them.  
\*/  
  
Log("\*\*\*\*\*\*\*\*\*\*\*\* ML\_Site\_Down Start NEW\*\*\*\*\*\*\*\*\*\*\*");  
  
log("Return ImpactFlag = 3 to prevent event reprocessing");  
// This have to be changed in order to be in sync with other policies or other  control  flag might be added  
@ImpactFlag = 3;   
ReturnEvent(EventContainer);   
  
// getting the site name  
site\_name = Substring(@Node, 0, 7);  
log(" Site Name : " + site\_name);  
  
// 0 - Is it a Clear Event? If Yes: Clear the synthetic events and EXIT. If No: Continue  
if (@Severity == 0)  
{  
    ExistingSiteDownDataType = "OS\_Status";  
    ExistingSiteDownFilter = "Node = '"+site\_name+"' And Summary = 'Site Down' And Severity = 5";  
    ExistingSiteDownOutput = GetByFilter(ExistingSiteDownDataType, ExistingSiteDownFilter,False);  
    NumExistingSiteDownOutput = Length(ExistingSiteDownOutput);  
      
    if (NumExistingSiteDownOutput > 0)  
    {  
        ExistingSiteDownOutput[0].Severity = 0;  
    }  
    exit();  
}  
// 1 - Enrich with site priority from CCMDB  
  
  
  
sleep\_time = 60;  
  
// getting the site level  
SiteLevelDataType = "site\_id";  
SiteLevelFilter = "SITE\_ID = '"+site\_name+"'";  
SitePriorityOutput = GetByFilter(SiteLevelDataType, SiteLevelFilter,False);  
NumSiteLevelOutput = Length(SitePriorityOutput);  
  
if (NumSiteLevelOutput > 0)  
{  
    site\_level = SitePriorityOutput[0].SITE\_LEVEL;  
      
    SleepTimeDataType = "SleepTime";   
    SleepTimeFilter = "SITE\_LEVEL = '"+site\_level+"'";  
    SleepTimeOutput = GetByFilter(SleepTimeDataType, SleepTimeFilter,False);  
    NumSleepTimeOutput = Length(SleepTimeOutput);  
      
    if (NumSleepTimeOutput > 0)  
    {  
        Log("\*\*\*\*\*\*\*\*\*\*\*\* Getting Sleep Time \*\*\*\*\*\*\*\*\*\*\*");  
        // checking whether it's day or night time  
        current\_time = getdate();  
        current\_date = LocalTime(current\_time, "MM/dd/yyyy");  
        start\_day\_time = "08:00:00 " + current\_date;  
        end\_day\_time = "18:00:00 " + current\_date;  
        start\_day\_time\_seconds = ParseDate(start\_day\_time, "HH:mm:ss MM/dd/yyyy");  
        end\_day\_time\_seconds = ParseDate(end\_day\_time, "HH:mm:ss MM/dd/yyyy");      
      
      
        if ( (current\_time > start\_day\_time\_seconds) and (current\_time < end\_day\_time\_seconds))  
        {  
            log ("It's Day Time");  
            sleep\_time = SleepTimeOutput[0].TIME\_DAY;  
        }  
        else  
        {  
            log ("It's Night Time");  
            sleep\_time = SleepTimeOutput[0].TIME\_NIGHT;  
        }  
    }  
    log (" Sleep Time :" + sleep\_time);  
}  
  
// 2 - Is there an existing TT for this Site Down?  
  
ExistingSiteDownDataType = "OS\_Status";  
ExistingSiteDownFilter = "Node = '"+site\_name+"' And Summary = 'Site Down' And Severity = 5";  
ExistingSiteDownOutput = GetByFilter(ExistingSiteDownDataType, ExistingSiteDownFilter,False);  
NumExistingSiteDownOutput = Length(ExistingSiteDownOutput);  
  
  
if (NumExistingSiteDownOutput > 0)  
{  
// 2.1 If yes: Update the TT by inserting the current event's details in the journal and EXIT  
    log ("\*\*\*\*\*\*\*\*\*\*\*\* TT has already created, updating the TT and EXIT \*\*\*\*\*\*\*\*\*\*\*");  
    JournalMessage = "Node: " + @Node + " EventType: " + @AlertGroup+ " Event Text: " + @AddText + " :: " + @AlertKey;  
    MobiLib.SiteDownInsertJournal (ExistingSiteDownOutput[0].ServerSerial, JournalMessage, @ServerSerial);  
          
    // ####### update LastOccurrence and Tally ??  
      
    exit();  
      
}  
else  
{  
// 3 - If No: Sleep using specific sleep time.  
  
    log ("\*\*\*\*\*\*\*\*\*\*\*\* TT has not created yet: Hibernate for: " +  sleep\_time + " seconds \*\*\*\*\*\*\*\*\*\*\*");  
    // Updating HibernateFlag and return event. HibernateFlag=1 (Hibernated)  
    // @HibernateFlag = 1;          
    // ReturnEvent(EventContainer);  
      
    //side\_down\_action\_key = @ServerSerial + @ServerName;  
    //Hibernate(side\_down\_action\_key, "Sleeping", sleep\_time);  
      
    // Updating HibernateFlag and return event. HibernateFlag=2 (Awake)  
    //@HibernateFlag = 2;          
    //ReturnEvent(EventContainer);  
      
// 3.1    Has the event cleared or is not in the O.S. anymore?  
  
    CurrentEventDataType = "OS\_Status";  
    CurrentEventFilter = "ServerSerial = "+@ServerSerial+"";  
    CurrentEventOut = GetByFilter(CurrentEventDataType, CurrentEventFilter,False);  
    NumCurrentEventOut = Length(CurrentEventOut);  
  
    if (NumCurrentEventOut == 0 OR CurrentEventOut[0].Severity == 0 )  
    {  
// 3.2 If yes: Exit  
      
        log ("\*\*\*\*\*\*\*\*\*\*\*\* The event has been cleared or  is not in the OS anymore: exiting \*\*\*\*\*\*\*\*\*\*\*");  
        exit();  
    }  
    else  
    {  
  
// 4 - If No: Has a synthetic been created?  
      
        ExistingSiteDownDataType = "OS\_Status";  
        ExistingSiteDownFilter = "Node = '"+site\_name+"' And Summary = 'Site Down' And Severity = 5";  
        ExistingSiteDownOutput = GetByFilter(ExistingSiteDownDataType, ExistingSiteDownFilter,False);  
        NumExistingSiteDownOutput = Length(ExistingSiteDownOutput);  
      
        if (NumExistingSiteDownOutput > 0)  
        {  
// 4.1    If Yes: Update the TT by inserting the current event's details in the journal and EXIT  
            log ("\*\*\*\*\*\*\*\*\*\*\*\* TT has already created, updating the TT and EXIT \*\*\*\*\*\*\*\*\*\*\*");  
            JournalMessage = "Node: " + @Node + " EventType: " + @AlertGroup+ " Event Text: " + @AddText;  
            MobiLib.SiteDownInsertJournal (ExistingSiteDownOutput[0].ServerSerial, JournalMessage, @ServerSerial);  
  
      
        // ####### update LastOccurrence and Tally ??  
      
            exit();  
        }  
        else  
        {  
// 5 - If No: Create synthetic event  
            log ("\*\*\*\*\*\*\*\*\*\*\*\* Create Synthetic Event: Site Down \*\*\*\*\*\*\*\*\*\*\*");  
              
            SiteDown = NewObject();  
      
            SiteDown.Node = site\_name;  
            SiteDown.Summary = "Site Down";  
            SiteDown.AlertGroup = @AlertGroup;  
            SiteDown.AlertKey = "Site Down";  
            SiteDown.Severity = 5;  
            SiteDown.Type = 1;  
            SiteDown.Agent = "Netcool Impact";  
            SiteDown.Manager = "Netcool Impact";  
            SiteDown.FirstOccurrence = getdate();  
            SiteDown.LastOccurrence = getdate();  
            SiteDown.LogTicket = 1;  
            SiteDown.Identifier = SiteDown.Node + SiteDown.Summary + SiteDown.Type;  
            SiteDown.Class = 200070; // Must be set properly  
            SiteDown.Domain = @Domain;  
            SiteDown.Region = @Region;  
            SiteDown.ManCity = @ManCity;  
  
            AddDataItem('OS\_Status', SiteDown);  
            Log("New Synthetic Site Down Event Created ");  
              
// 5.1    Add details of all associated environmental events and clear them.  
              
            // Searching for Environmental Alarms  
          
            ExistingEnvDataType = "OS\_Status";  
            ExistingEnvFilter = "Node like '"+site\_name+"' And (EventType = 'environmentalAlarm' OR EventType = 'Environmental Alarm') And Severity > 1";  
            ExistingEnvOutput = GetByFilter(ExistingEnvDataType, ExistingEnvFilter ,False);  
            NumExistingEnvOutput = Length(ExistingEnvOutput);  
            log(" NumExistingEnvOutput = " + NumExistingEnvOutput);  
          
            if (NumExistingEnvOutput > 0)  
            {  
              
                counter = 0;  
                log("counter = " + counter);  
                  
                // Get the ServerSerial and ServerName of the Site Down event just created  
                ExistingSiteDownDataType = "OS\_Status";  
                ExistingSiteDownFilter = "Node = '"+site\_name+"' And Summary = 'Site Down' And Severity = 5";  
                ExistingSiteDownOutput = GetByFilter(ExistingSiteDownDataType, ExistingSiteDownFilter,False);  
                NumExistingSiteDownOutput = Length(ExistingSiteDownOutput);  
                  
                while (counter < NumExistingEnvOutput)  
                {  
                    // Clear the Environmental events and add their details in the synthetic event.  
                    log("counter = " + counter);  
                    ExistingEnvOutput[counter].Severity = 0;  
                    // Insert Env events details in the "Site Down" event.  
                    journal\_node = ExistingEnvOutput[counter].Node;  
                    journal\_event\_type = ExistingEnvOutput[counter].EventType;  
                    // journal\_add\_tex = ExistingEnvOutput[counter].AddText;  
                    site\_down\_serial = ExistingSiteDownOutput[0].ServerSerial;  
                    // site\_down\_server = ExistingSiteDownOutput[0].ServerName;  
                    JournalMessage = "Node: " + journal\_node  + " EventType: " + journal\_event\_type + " Event AlertKey: " + @AlertKey;  
                    log("JournalInsert = " + JournalMessage);  
                    MobiLib.SiteDownInsertJournal (site\_down\_serial, JournalMessage, ExistingEnvOutput[counter].ServerSerial);  
  
                    counter = counter + 1;  
                }  
            }  
        }  
    }  
}