Managing Reports

Accessing Reports

You can access reports using CloupiaScript. You can use the report data to make dynamic decisions for subsequent tasks.

For example, to allocate an unassociated Cisco UCS B-Series Blade Server that is greater than 32GB, use the following script to query the list of all Cisco UCS servers that are managed by a specific Cisco UCS Manager. The script shows how to filter a subset of values selectively. The `getReportView(reportContext, reportName)` function takes reportContext and reportName as arguments and returns the TableView object which displays the content in a table format.

```
importPackage(java.lang);
importPackage(java.util);
importPackage(com.cloupia.lib.util.managedreports);

function getReport(reportContext, reportName)
{
     var report = null;
      try
      {

             report = ctxt.getAPI().getConfigTableReport(reportContext, reportName);
      } catch(e)
      {
      }

      if (report == null)
      {
             return ctxt.getAPI().getTabularReport(reportName, reportContext);
      } else
     {
           var source = report.getSourceReport();
           return ctxt.getAPI().getTabularReport(source, reportContext);
     }
}

function getReportView(reportContext, reportName)
{
      var report = getReport(reportContext, reportName);
```

```
     if (report == null)
     {
           logger.addError("No such report exists for the specified context "+reportName);

           return null;
     }

     return new TableView(report);
}

// following are only sample values and need to be modified based on actual UCSM account
name
var ucsmAccountName = "ucs-account-1";

// report name is obtained from Report Meta. No need to change unless you need to access a
 different report
var reportName = "UcsController.allservers.table_config";

var repContext = util.createContext("ucsm", null, ucsmAccountName);
// Enable Developer Menu in UCSD and find reportName in the Report Metadata for the specific
 report
// Creating a ReportContext
// @param contextName
// Refer to UCSD API Guide for the available contexts
// @param cloud
// should be null unless contextName is "cloud" or "host node"

// @param value
// identifier of the object that is going to be referenced

Report var report = getReportView(repContext, reportName);

// Get only the rows for which Server Type column value is B-Series
report = report.filterRowsByColumn("Server Type", "B-Series", false);

// now look for unassociated servers only
report = report.filterRowsByColumn("Operation State", "unassociated", false);

// Make sure servers are actually in available state
report = report.filterRowsByColumn("Availability", "available", false);

var matchingIds = [];
var count = 0;


// Now look for Servers with memory of 32 GB or more
for (var i=0; i<report.rowCount(); i++)
{

      var memory = Integer.parseInt(report.getColumnValue(i, "Total Memory (MB)"));

      logger.addDebug("Possible Server "+report.getColumnValue(i, "ID")+", mem="+memory);

      if (memory >= 32*1024)
      {
               matchingIds[count++] = report.getColumnValue(i, "ID");
      }
}

if (count == 0)
{
     ctxt.setFailed("No servers matched the criteria");
     ctxt.exit();
}
```

```
// Now randomly pick one of the item from the filtered list
var id = matchingIds[Math.round(Math.random()*count)];
logger.addInfo("Allocated server "+id);

// Save the Server-ID to the global inputs
ctxt.updateInput("SELECTED_UCS_SERVER_ID", id);
```

Accessing Tabular Reports

If you use the `getTabularReport(reportName, reportContext)` API to access a tabular report, you can view the report details in the user interface (UI) in one of the following ways:

- Reports Customization—To access the reports customization tab, choose **Administration** > **User Interface Settings** and click **Reports Customization**. The customization report displays the report details such as menu, context, report type, and so on. To customize the table columns, click the down arrow that appears on the column header when you place the cursor. From the drop-down menu, choose **Cloumns** and check the columns to be shown. For example, to display the report ID, check **ID**.
- Report Metadata—The report metadata link appears in the UI only when the developer menu is enabled.

Some of the important report details are:

- API report ID—You can use the API report ID column to get the value for the reportID parameter to use in the REST URL when you use the userAPIGetTabularReport API. This REST API is used to retrieve the tabular report from a web browser or other REST client application.
- ID—The ID column displays the report name. You can use the ID column to get the reportName parameter when you use the getTabularReport API in CloupiaScript. This parameter is also applicable for the getConfigTableReport API.
- context—To construct the ReportContext, you need two input parameters: contextName and contextValue. For regular contexts, use `util.createContext ("contextName", null, "instanceName")`. For example, `util.createContext("vm",null, vmId)`, where vmId is the integer VM ID value to uniquely identify a VM in UCS Director. For cloud contexts, use `util.createContext("contextName","cloudInstanceName", null)` , or `util.createContext("contextName",null,"cloudInstanceName")`. For example, `util.createContext("cloud","All Clouds",null)` , or `util.createContext("cloud", null, "All Clouds")`.

Emailing Reports

You can use CloupiaScript to email a report to a user. To email this report periodically, set up a workflow schedule for this workflow at the desired frequency.

The following script enables user to choose a report name from the report list and email the report to the specified email address.

```
importPackage(java.util);
importPackage(java.lang);
importPackage(java.io);
importPackage(com.cloupia.model.cEvent.notify);
importPackage(com.cloupia.model.cIM);
importPackage(com.cloupia.lib.util.mail);
importPackage(com.cloupia.fw.objstore);
importPackage(com.cloupia.lib.util.managedreports);
```

```
importPackage(com.cloupia.lib.util);
importPackage(com.cloupia.service.cIM.inframgr);
importPackage(org.apache.commons.httpclient);
importPackage(org.apache.commons.httpclient.cookie);
importPackage(org.apache.commons.httpclient.methods);
importPackage(org.apache.commons.httpclient.auth);
importPackage(com.cloupia.model.cEvent.notify);

function getMailSettings()
{
      return ObjStoreHelper.getStore((new MailSettings()).getClass()).getSingleton();
}
function getReport(reportContext, reportName)
{
     var report = null;
      try
      {
             report = ctxt.getAPI().getConfigTableReport(reportContext, reportName);
      } catch(e)

      {
      }
      if (report == null)
      {
             return ctxt.getAPI().getTabularReport(reportName, reportContext);
      } else
     {
           var source = report.getSourceReport();
           return ctxt.getAPI().getTabularReport(source, reportContext);
     }
}
function getReportView(reportContext, reportName)
{
      var report = getReport(reportContext, reportName);
     if (report == null)
     {
           logger.addError("No such report exists for the specified context "+reportName);
           return null;
     }
     return new TableView(report);
}
var ucsmAccountName = ctxt.getInput("Account_Name");
var reportName = ctxt.getInput("REPORT_NAME");

var reportContextType=ReportContext.getDynamicContextLevel("apic_controller");
var repContext = util.createContextByType(reportContextType, null, ucsmAccountName);

var report = getReportView(repContext, reportName);
logger.addInfo("Report Context Type is :::::::::: " + reportContextType);
//var repContext = util.createContext("apic_controller", null, ucsmAccountName);
//var report = getReportView(repContext, reportName);
var numRowsFound = report.rowCount();
logger.addInfo("Number of Rows found: " + numRowsFound);
var toEmail = [ ctxt.getInput("Email Address") ];
var message = new EmailMessageRequest();

message.setToAddrs(toEmail);
message.setSubject("APIC Report : "+ ucsmAccountName);
message.setFromAddress("no-reply@cisco.com");
var buffer = new StringWriter();
var printer = new PrintWriter(buffer);
var formatter = new com.cloupia.lib.util.managedreports.Formatter(new File("."), printer);

formatter.printTable(report);
```

```
printer.close();
var body = "<head><style type='text/css'>";
body = body + "table { font-family: Verdana, Geneva, sans-serif; font-size: 12px; border:
thin solid #039; border-spacing: 0; background: #ffffff; } ";
body = body + " th { background-color: #6699FF; color: white; font-family: Verdana, Geneva,
 sans-serif; font-size: 10px; font-weight: bold; border-color: #CCF; border-style: solid;
border-width: 1px 1px 0 0; margin: 0; padding: 5px; } ";
body = body + " td { font-family: Verdana, Geneva, sans-serif; font-size: 10px; border-color:
 #CCF; border-style: solid; border-width: 1px 1px 0 0; margin: 0; padding: 5px; background:
 #ffffff; }";
body = body + "</style></head>";
body = body+ "<body><h1>APIC Report</h1><br>" + buffer.toString();

message.setMessageBody(body);
logger.addInfo("Sending email");
MailManager.sendEmail("APIC Report", getMailSettings(), message);
```