This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2013 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.
Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.
## Contents

### Chapter 1: Introduction

<table>
<thead>
<tr>
<th>Intended Audience</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>About the XML Open Gateway</td>
<td>16</td>
</tr>
<tr>
<td>The XOG Client</td>
<td>16</td>
</tr>
<tr>
<td>The XOG Web Services</td>
<td>17</td>
</tr>
<tr>
<td>Web Services Descriptive Language (WSDL)</td>
<td>17</td>
</tr>
<tr>
<td>XOG Access Rights</td>
<td>18</td>
</tr>
</tbody>
</table>

### Chapter 2: Installing the XOG Client

<table>
<thead>
<tr>
<th>How to Install the XOG Client</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Installation</td>
<td>22</td>
</tr>
<tr>
<td>Cross-Platform Installation</td>
<td>22</td>
</tr>
<tr>
<td>Verify the XOG Client Version</td>
<td>23</td>
</tr>
<tr>
<td>FIPS 140-2 Mode Setup</td>
<td>24</td>
</tr>
<tr>
<td>XOG Client Directories</td>
<td>25</td>
</tr>
</tbody>
</table>

### Chapter 3: Running the XOG

<table>
<thead>
<tr>
<th>How to Run the XOG</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the XOG from the Command Line</td>
<td>27</td>
</tr>
<tr>
<td>Command Line Parameters (XOG Client)</td>
<td>28</td>
</tr>
<tr>
<td>About the .properties File</td>
<td>30</td>
</tr>
<tr>
<td>Create a .properties File</td>
<td>31</td>
</tr>
<tr>
<td>Run the XOG Using the .properties File</td>
<td>32</td>
</tr>
<tr>
<td>Run the XOG Client as a Shell</td>
<td>32</td>
</tr>
<tr>
<td>XOG Client Shell Commands</td>
<td>33</td>
</tr>
<tr>
<td>How to Run the XOG Using SOAP</td>
<td>34</td>
</tr>
</tbody>
</table>

### Chapter 4: Usage Guidelines

<table>
<thead>
<tr>
<th>About the Schema Files</th>
<th>37</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema Definitions</td>
<td>38</td>
</tr>
<tr>
<td>NikuDataBus Header Element</td>
<td>39</td>
</tr>
<tr>
<td>Attribute Information in the Schema</td>
<td>40</td>
</tr>
<tr>
<td>About the XML Read and Write Files</td>
<td>40</td>
</tr>
<tr>
<td>What is in an XML Read File</td>
<td>41</td>
</tr>
<tr>
<td>How to Create an XML Write File</td>
<td>42</td>
</tr>
</tbody>
</table>
Chapter 5: XOG Services

Object API ......................................................................................................................... 53
  XOG Object Types ........................................................................................................... 53
  ActionObject Element ..................................................................................................... 54
  Read Standard Stock Objects ......................................................................................... 55
  Write Standard Stock Objects ....................................................................................... 56
  Partitioning and Standard Stock Objects ....................................................................... 58
  Custom Object Instances ............................................................................................... 59
  ContentPack Service ....................................................................................................... 60
  Autonumbering and Custom Attributes .......................................................................... 66
  Import and Export Custom Fiscal Time-Varying Attributes ............................................ 66
  Import and Export UI Themes ......................................................................................... 67
  Include New and Delete Buttons .................................................................................... 68
  About Passing XDM Custom Fields ................................................................................. 68

InvokeAction API .............................................................................................................. 68
  InvokeAction API Root Elements ................................................................................... 69
  FlushCache Elements ...................................................................................................... 69
  Process Elements ............................................................................................................ 70

Query API ........................................................................................................................... 71
  Query API Root Elements ............................................................................................... 72
  The Query Filter ............................................................................................................... 72
  Example: Exporting Query Results to a Tab-Delimited Text File ..................................... 75

Chapter 6: GEL Scripting

GEL Overview ..................................................................................................................... 77
GEL Setup ............................................................................................................................ 78
GEL Script Validation and Execution ............................................................................... 78
GEL Basics .......................................................................................................................... 78
  GEL Script Structure ....................................................................................................... 79
  GEL Script Tags ............................................................................................................... 79
  Conditionals and Loops ................................................................................................... 80
  Variables .......................................................................................................................... 81
  Built-in Parameters .......................................................................................................... 82
  Things to Watch For When Using GEL .......................................................................... 83
  Using SSL with GEL ......................................................................................................... 83
  Set GEL Tag Restrictions .............................................................................................. 84
Examples for Running the XOG ................................................................. 85
  Example 1: Individual Calls .................................................................. 86
  Example 2: Single Invocation .................................................................. 88
Database Operations .................................................................................. 89
File Operations ......................................................................................... 91
  Example 1: Create a Rate Matrix XOG File ........................................... 92
  Example 2: Output Delimited Files ....................................................... 93
  Example 3: Create a File to Write in Documents for Multiple Projects .... 93
Integration Processes ............................................................................... 96
  Basic Integration Process Checklist ...................................................... 97

Chapter 7: XOG WSDL ................................................................. 99

About the WSDL ..................................................................................... 99
  Access the WSDL .................................................................................. 99
  Viewers for WSDL ............................................................................... 100
  Emitter Tools for the WDSL ................................................................. 101
Object WSDL ......................................................................................... 101
  InvokeAction WSDL ............................................................................ 101
    Example: Process WSDL .................................................................... 102
    <xsd:any> and Process WSDL ............................................................ 105
Query WSDL ......................................................................................... 105
  Example: Demand for Resource Query WSDL ..................................... 106
Examples: Microsoft Visual Studio (.NET) ............................................... 110
  Generate Supporting API ..................................................................... 110
    Generate a Proxy API from Axis ......................................................... 111
    Add a Web Reference from Microsoft Visual Studio ......................... 111

Appendix A: XOG Object Reference .................................................... 113

Stock XOG Object Summary .................................................................. 113
  Base XOG Objects ............................................................................... 113
  Product Stock XOG Object Summary .................................................. 114
Application ......................................................................................... 116
Asset ..................................................................................................... 117
Benefit Plan ......................................................................................... 118
  BenefitPlan Schema Tag ...................................................................... 119
  Description Schema Tag ...................................................................... 121
Detail Schema Tag ............................................................................... 121
  Segment Schema Tag .......................................................................... 122
Budget Plan .......................................................................................... 123
  BudgetPlan Schema Tag ...................................................................... 125
  Description Schema Tag ...................................................................... 127
<table>
<thead>
<tr>
<th>Tag</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grouping Attributes Schema Tag</td>
<td>128</td>
</tr>
<tr>
<td>Detail Schema Tag</td>
<td>129</td>
</tr>
<tr>
<td>Segment Schema Tag</td>
<td>129</td>
</tr>
<tr>
<td>Capacity Planning Scenario</td>
<td>131</td>
</tr>
<tr>
<td>Scenario Schema Tag</td>
<td>132</td>
</tr>
<tr>
<td>Segment Schema Tag</td>
<td>136</td>
</tr>
<tr>
<td>Change Request</td>
<td>140</td>
</tr>
<tr>
<td>Change Request Schema Tag</td>
<td>141</td>
</tr>
<tr>
<td>Charge Code</td>
<td>145</td>
</tr>
<tr>
<td>Charge Code (chargecode) Schema Tag</td>
<td>146</td>
</tr>
<tr>
<td>Company</td>
<td>147</td>
</tr>
<tr>
<td>Company Schema Tag</td>
<td>150</td>
</tr>
<tr>
<td>Contact Information Schema Tag</td>
<td>151</td>
</tr>
<tr>
<td>Supplemental Information Schema Tag</td>
<td>153</td>
</tr>
<tr>
<td>Custom Information Schema Tag</td>
<td>155</td>
</tr>
<tr>
<td>Financial Information Schema Tag</td>
<td>156</td>
</tr>
<tr>
<td>Billing Address Schema Tag</td>
<td>158</td>
</tr>
<tr>
<td>Billing Address Detail Schema Tag</td>
<td>158</td>
</tr>
<tr>
<td>OBS Associations Schema Tag</td>
<td>160</td>
</tr>
<tr>
<td>Company Class</td>
<td>161</td>
</tr>
<tr>
<td>Company Class Schema Tag</td>
<td>162</td>
</tr>
<tr>
<td>Content Pack</td>
<td>163</td>
</tr>
<tr>
<td>Filter Mapping (filterMapping) Schema Tag</td>
<td>170</td>
</tr>
<tr>
<td>portlet Schema Tag</td>
<td>170</td>
</tr>
<tr>
<td>Cost Plan</td>
<td>176</td>
</tr>
<tr>
<td>CostPlan Schema Tag</td>
<td>177</td>
</tr>
<tr>
<td>Description Schema Tag</td>
<td>179</td>
</tr>
<tr>
<td>Grouping Attributes Schema Tag</td>
<td>179</td>
</tr>
<tr>
<td>Detail Schema Tag</td>
<td>180</td>
</tr>
<tr>
<td>Segment Schema Tag</td>
<td>180</td>
</tr>
<tr>
<td>Cost Plus Code</td>
<td>182</td>
</tr>
<tr>
<td>Cost Plus Rule (costplusrule) Schema Tag</td>
<td>183</td>
</tr>
<tr>
<td>Department</td>
<td>184</td>
</tr>
<tr>
<td>Departments Schema Tag</td>
<td>185</td>
</tr>
<tr>
<td>Description Schema Tag</td>
<td>186</td>
</tr>
<tr>
<td>LocationAssociations Schema Tag</td>
<td>187</td>
</tr>
<tr>
<td>Department (Child Department) Schema Tag</td>
<td>189</td>
</tr>
<tr>
<td>obsTypes Schema Tag</td>
<td>189</td>
</tr>
<tr>
<td>Entity</td>
<td>191</td>
</tr>
<tr>
<td>Entity Schema Tag</td>
<td>192</td>
</tr>
<tr>
<td>Description Schema Tag</td>
<td>195</td>
</tr>
<tr>
<td>Short Description (shortDescription) Schema Tag</td>
<td>195</td>
</tr>
<tr>
<td>Schema Tag Name</td>
<td>Page</td>
</tr>
<tr>
<td>----------------</td>
<td>------</td>
</tr>
<tr>
<td>OBS Types (obsTypes)</td>
<td>196</td>
</tr>
<tr>
<td>GL Periods [GLPeriods]</td>
<td>198</td>
</tr>
<tr>
<td>Plan Defaults [PlanDefaults]</td>
<td>199</td>
</tr>
<tr>
<td>Grouping Attributes</td>
<td>200</td>
</tr>
<tr>
<td>Financial Transaction</td>
<td>201</td>
</tr>
<tr>
<td>Transactions Schema Tag</td>
<td>202</td>
</tr>
<tr>
<td>Transaction Import Schema Tag</td>
<td>207</td>
</tr>
<tr>
<td>General Ledger Account</td>
<td>210</td>
</tr>
<tr>
<td>GLAccount Schema Tag</td>
<td>211</td>
</tr>
<tr>
<td>General Ledger Allocation Rule</td>
<td>214</td>
</tr>
<tr>
<td>GL Allocation Rule Schema Tag</td>
<td>215</td>
</tr>
<tr>
<td>General Ledger Period</td>
<td>218</td>
</tr>
<tr>
<td>GLPeriod Schema Tag</td>
<td>219</td>
</tr>
<tr>
<td>General Ledger Transaction</td>
<td>220</td>
</tr>
<tr>
<td>GLTransaction Schema Tag</td>
<td>222</td>
</tr>
<tr>
<td>Idea</td>
<td>224</td>
</tr>
<tr>
<td>Idea Schema Tag</td>
<td>225</td>
</tr>
<tr>
<td>Investments Schema Tag</td>
<td>228</td>
</tr>
<tr>
<td>Budget Schema Tag</td>
<td>231</td>
</tr>
<tr>
<td>Inbound Transaction</td>
<td>238</td>
</tr>
<tr>
<td>inboundTransactionType Schema Tag</td>
<td>239</td>
</tr>
<tr>
<td>Incident</td>
<td>241</td>
</tr>
<tr>
<td>incidents Schema Tag</td>
<td>242</td>
</tr>
<tr>
<td>incidents Schema Tag</td>
<td>243</td>
</tr>
<tr>
<td>Investment</td>
<td>245</td>
</tr>
<tr>
<td>Investment Schema Tag</td>
<td>249</td>
</tr>
<tr>
<td>Allocations Schema Tag</td>
<td>262</td>
</tr>
<tr>
<td>Details Schema Tag</td>
<td>264</td>
</tr>
<tr>
<td>Financial Transaction (scenarioDependencies) Schema Tag</td>
<td>265</td>
</tr>
<tr>
<td>InvestmentAssociations Schema Tag</td>
<td>266</td>
</tr>
<tr>
<td>InvestmentBaselines Schema Tag</td>
<td>267</td>
</tr>
<tr>
<td>UsageCurve and CostCurve Schema Tags</td>
<td>269</td>
</tr>
<tr>
<td>InvestmentResources Schema Tag</td>
<td>269</td>
</tr>
<tr>
<td>InvestmentTasks Schema Tag</td>
<td>273</td>
</tr>
<tr>
<td>General Schema Tag</td>
<td>278</td>
</tr>
<tr>
<td>OBSAssocs Schema Tag</td>
<td>278</td>
</tr>
<tr>
<td>Custom Information Schema Tag</td>
<td>279</td>
</tr>
<tr>
<td>Issue</td>
<td>280</td>
</tr>
<tr>
<td>Issue Schema Tag</td>
<td>282</td>
</tr>
<tr>
<td>Location</td>
<td>284</td>
</tr>
<tr>
<td>Locations Schema Tag</td>
<td>285</td>
</tr>
<tr>
<td>Description Schema Tag</td>
<td>287</td>
</tr>
</tbody>
</table>
DepartmentAssociations Schema Tag ......................................................... 288
Child Location Schema Tag ...................................................................... 288
Matrix ........................................................................................................ 289
columnType Schema Tag ........................................................................... 290
MatrixRowType Schema Tag ..................................................................... 292
Non-Project Investment .............................................................................. 295
Asset Schema Tag ..................................................................................... 297
Application Schema Tag ............................................................................ 298
Product Schema Tag .................................................................................. 300
Other Work Schema Tag ............................................................................ 300
NPIO Common Fields Schema Tag ............................................................. 301
Service Schema Tag .................................................................................. 312
Notifications ............................................................................................. 313
Notification Schema Tag ........................................................................... 314
Portfolio ....................................................................................................... 315
Portfolio Schema Tag .................................................................................. 315
Contents Schema Tag .................................................................................. 319
Process Notification .................................................................................... 321
Process Notification Schema Tag ................................................................. 322
Project ........................................................................................................ 324
SRM_PROJ ECTS Schema Tag .................................................................... 327
SRM_RESOURCES Schema Tag .................................................................... 329
Project (PRI_PROJECTS) Schema Tag ......................................................... 329
Project (PAC_MNT_PROJECTS) Schema Tag .............................................. 330
Project Location Schema Tag .................................................................... 331
Project (CLNTSUPP) Schema Tag ............................................................. 332
Resource (PRTEAM) Schema Tag .............................................................. 332
SRM_RESOURCES Schema Tag .................................................................... 333
SRM_RESOURCES Schema Tag .................................................................... 333
SRM_RESOURCES Schema Tag .................................................................... 333
SRM_RESOURCES Schema Tag .................................................................... 334
Task (PRTask) Schema Tag ........................................................................ 334
TaskLabor (PRAssignment) Schema Tag .................................................... 337
OBS Association Schema Tag .................................................................... 338
OBS Association (OBSAssoc) Schema Tag .............................................. 338
Project (PRI_PROJECTS) Schema Tag ......................................................... 339
PAC_MNT_PROJECTS Schema Tag ............................................................. 344
Resource (PRTeam) Schema Tag ............................................................... 349
PRTask Schema Tag ................................................................................... 351
TaskLabor (PRAssignment) Schema Tag .................................................... 355
Requirement .............................................................................................. 359
RQP_RELEASES Schema Tag ...................................................................... 361
RQP_REQUIREMENTS Schema Tag ............................................................ 361
RQP_REQ_DEPENDENCIES Schema Tag ................................................................. 364
PRTASK Schema Tag ......................................................................................... 365
SRM_RESOURCES Schema Tag ........................................................................ 365
INV_INVESTMENTS Schema Tag ...................................................................... 366
Resource ............................................................................................................ 366
  Personal Information Schema Tag ................................................................ 371
  Contact Information Schema Tag (Resources XOG) ....................................... 373
  Management Information Schema Tag ......................................................... 375
  Financial Information Schema Tag ............................................................... 378
  Expenses Schema Tag .................................................................................. 379
  Rates and Costs Schema Tag ....................................................................... 380
  Custom Information Schema Tag (Resources XOG) ..................................... 380
  OBS Associations Schema Tag (Resources XOG) ........................................ 381
  SkillAssocs Schema Tag (Resources XOG) .................................................... 382
Resource Class .................................................................................................. 384
  Resource Class (resourceclass) Schema Tag ............................................. 385
Risk ...................................................................................................................... 386
  Risk Schema Tag .......................................................................................... 388
Role ...................................................................................................................... 391
  PRJ_RESOURCES Schema Tag .................................................................... 393
  SRM_RESOURCES Schema Tag ................................................................... 394
  PRJ_ROLES_FLAT Schema Tag .................................................................... 394
  Financial Information Schema Tag ............................................................... 395
  SkillAssocs Schema Tag ................................................................................ 396
Skill ....................................................................................................................... 397
  Skill Schema Tag .......................................................................................... 398
Subproject (Program) ......................................................................................... 399
  PRSubproject Schema Tag (Inbound and Outbound) ................................... 400
  PRSubproject Schema Tag (Inbound only) ................................................... 401
Subscription ...................................................................................................... 402
  Subscription Schema Tag ............................................................................. 403
Time Period .......................................................................................................... 406
  TimePeriod Schema Tag ................................................................................ 407
  PRTimesheet Schema Tag ............................................................................. 408
  SRM_RESOURCES Schema Tag ................................................................... 409
  TimePeriod Schema Tag ................................................................................ 409
  PRTimesheet Schema Tag ............................................................................. 409
  PRTimesheet Schema Tag ............................................................................. 409
  PRTimesheet Schema Tag ............................................................................. 411
  PRTimesheet Schema Tag ............................................................................. 411
  PRJ_Projects Schema Tag .......................................................................... 412
  PRTask Schema Tag ...................................................................................... 412
  PRAssignments Schema Tag ......................................................................... 413
Appendix C: GEL Tag Library Reference

Tag Libraries .................................................................................................................. 449
GEL Tag Library ............................................................................................................. 449
gel:script - Defining GEL Scripts .................................................................................. 450
gel:parse - Parsing XML ................................................................................................ 451
gel:set - Setting XML Document Values ....................................................................... 452
gel:expr - Evaluating Expressions ................................................................................ 456
gel:parameter - Defining Parameters .............................................................................. 457
gel:getDocument - Requesting XML Documents ............................................................ 459
gel:setDocument - Passing XML Documents .................................................................. 459
gel:persist - Persisting Variables .................................................................................. 460
<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>gel:notify</td>
<td>Sending Notifications</td>
<td>462</td>
</tr>
<tr>
<td>gel:email</td>
<td>Sending Email Messages</td>
<td>463</td>
</tr>
<tr>
<td>gel:formatDate</td>
<td>Formatting Time Strings</td>
<td>464</td>
</tr>
<tr>
<td>gel:parseDate</td>
<td>Parsing Time Strings</td>
<td>466</td>
</tr>
<tr>
<td>gel:setDataSource</td>
<td>Specifying Data Sources</td>
<td>467</td>
</tr>
<tr>
<td>gel:nsqlQuery</td>
<td>Executing NSQL Queries</td>
<td>467</td>
</tr>
<tr>
<td>gel:log</td>
<td>Logging Messages</td>
<td>468</td>
</tr>
<tr>
<td>gel:out</td>
<td>Printing to the Console</td>
<td>469</td>
</tr>
<tr>
<td>Core Tag Library</td>
<td></td>
<td>470</td>
</tr>
<tr>
<td>core:catch</td>
<td>Catching Exceptions</td>
<td>471</td>
</tr>
<tr>
<td>core:set</td>
<td>Setting Variables</td>
<td>471</td>
</tr>
<tr>
<td>core:forEach</td>
<td>Iterating over Elements</td>
<td>474</td>
</tr>
<tr>
<td>core:if</td>
<td>Evaluating Conditionally</td>
<td>476</td>
</tr>
<tr>
<td>SOAP Tag Library</td>
<td></td>
<td>477</td>
</tr>
<tr>
<td>soap:invoke</td>
<td>Invoking SOAP Web Services</td>
<td>478</td>
</tr>
<tr>
<td>soap:envelope</td>
<td>Generating a SOAP Envelope</td>
<td>478</td>
</tr>
<tr>
<td>soap:header</td>
<td>Specifying the SOAP Header</td>
<td>479</td>
</tr>
<tr>
<td>soap:body</td>
<td>Specifying the SOAP Body</td>
<td>480</td>
</tr>
<tr>
<td>soap:attachment</td>
<td>Attaching Files to SOAP Requests</td>
<td>482</td>
</tr>
<tr>
<td>soap:message</td>
<td>Specifying SOAP XML Messages</td>
<td>482</td>
</tr>
<tr>
<td>Example: XOG Login and Read Objects Example</td>
<td></td>
<td>483</td>
</tr>
<tr>
<td>Example: Execute External Web Services with Attachments</td>
<td></td>
<td>485</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

This section contains the following topics:

- **Intended Audience** (see page 15)
- **About the XML Open Gateway** (see page 16)
- **The XOG Client** (see page 16)
- **The XOG Web Services** (see page 17)
- **Web Services Descriptive Language (WSDL)** (see page 17)
- **XOG Access Rights** (see page 18)

**Intended Audience**

Welcome to the *CA Clarity Integration Guide*. This document contains technical information needed to work with the XML Open Gateway (XOG).

**Intended Audience**

The audience for this guide includes integrators or system administrators who have the need to read data from or write data to CA Clarity using XML and web services. This guide assumes the reader is already familiar with XML code and the CA Clarity application.

**Document Contents**

This guide contains task oriented, conceptual, and reference material. The appendixes contain reference material on the following:

- Objects that can be read from or written to CA Clarity using the XOG
- GEL tags that can be used with XML for more advanced custom integration tasks
About the XML Open Gateway

The XML Open Gateway (XOG) is a CA Clarity web service interface that you can use to:

- Import data
- Export data
- Move configuration data from one system to another

CA Clarity web services are available on the same HTTP or HTTPS port as the HTML web browser interface. You can access a web service using one of the following:

- **XOG client**
  
  You can download the XOG client to your computer and use it to run the XOG.

- **Simple Object Access Protocol (SOAP)**
  
  You can access and run the XOG directly using SOAP without using the XOG client. CA Clarity web services use XML messages that follow the SOAP standard.

The XOG Client

The XOG client is a Java program that you can install on your computer and use to import and export data using the XOG. The XOG client communicates with the CA Clarity server on the standard HTTP port using the SOAP protocol. Using the client, you can:

- Log in to start an authenticated session
- Execute requests to read or write CA Clarity data
- Log out to end the session
The XOG Web Services

The following web services are available from the XOG API:

Object API
This API includes all read and write services for objects whose data can be imported or exported using the XOG.

InvokeAction API
This API provides for administrative actions that fall outside the categories of data import and export. It contains two root elements:

- FlushCache
- Process

Query API
This API lets you execute NSQL-based queries from the XOG. You can use this API to select and export the exact information you need from CA Clarity.

Web Services Descriptive Language (WSDL)

The Web Services Description Language (WSDL) describes the available XOG services and indicates how to communicate with the services. WSDL is used with SOAP and the XML schema to provide web services over the internet. You can connect to a web service and read the appropriate WSDL file to learn what functions are available on the server.
XOG Access Rights

Access Rights to Run the XOG from the Client

Before using the XOG client, you must have a valid CA Clarity login name and password.

You must also have one of the following access rights:

- Administration - Access
- Administration - XOG

XOG Access Rights for Individual Objects

Before a resource can use the XOG to import or export data for a particular object, you must assign the resource the XOG access right for that object (for example, Asset - XOG Access, Project - XOG Access, Resource - XOG Access, and so on).

For example, you can grant the Asset - XOG Access right to a resource to support a custom CA Clarity PPM desktop application that needs asset information. While the resource can import and export instance data that is associated with the asset object, the resource is not able to import or export data on any other objects.

XOG access rights for objects are listed in the access rights list in the Administration Tool with other access rights. XOG access rights are global rights.
To assign XOG access right to a resource

1. Click the Administration menu, and select Resources from the Organization and Access menu.
   
   The list page appears.

2. Click a name.
   
   The properties page appears.

3. Click Resource's Access Rights and go to Global.
   
   The access rights list page appears.

4. Click Add.
   
   The access rights selection page appears.

5. Enter *XOG Access in the Access Right field and click Filter.
   
   A list of XOG access rights for individual objects appears.

6. Select the appropriate XOG access rights and click Add.
   
   The XOG access right appears in the list of access rights for the resource.

7. Click Exit when you are done.

For more information, see the Administration Guide.
Chapter 2: Installing the XOG Client

This section contains the following topics:

- How to Install the XOG Client (see page 21)
- Windows Installation (see page 22)
- Cross-Platform Installation (see page 22)
- Verify the XOG Client Version (see page 23)
- FIPS 140-2 Mode Setup (see page 24)
- XOG Client Directories (see page 25)

How to Install the XOG Client

Use the following process to install the XOG client.

1. Download and install one of the following XOG client versions on your computer:
   - Windows XOG Client executable
   - Cross-platform ZIP archive (for non-Windows platforms) (see page 22)

2. Verify that the XOG client version matches the version of CA Clarity it is to work with.

3. Set up the Java Runtime Environment (JRE).

4. (Optional) Set up FIPS 140-2 mode if this standard is required for your business.
   - This mode is a standard that describes the U.S. federal government requirements for encrypting sensitive data.
Windows Installation

If your computer is running on a Windows platform, use the following instructions to download and install the XOG client.

To install the XOG client for Windows
1. Log in to CA Clarity.
2. Click the Administration menu and select Client Downloads from the General Settings menu.
   The client downloads page appears.
3. Click Download for the Windows Installer.
   The download dialog box appears.
4. Click Save File and save the XOG.exe file to a directory on your local computer.
5. On your computer, run XOG.exe and follow the instructions that appear on the screen.

Cross-Platform Installation

If your computer is running on a non-Windows platform, use the following instructions to download and install the XOG client.

To install the cross-platform XOG client
1. Log in to CA Clarity.
2. Click the Administration menu, and select Client Downloads from the General Settings menu.
   The client downloads page appears.
3. Click Download for the Cross-platform ZIP.
   The File Download dialog box appears.
4. Click Save and save the xogclient.zip file to your local computer.
5. Create a local folder named xogclient and extract the xogclient.zip files to the folder.
6. (UNIX only) From the bin directory, run the following command:
   chmod +x run.sh
Verify the XOG Client Version

Over time, a mismatch between the application and the XOG client can occur if the application is upgraded and the XOG client is not. Verify that the XOG client version you are using matches the version of CA Clarity PPM you are using. If the version numbers do not match, download the XOG client from CA Clarity PPM and reinstall it.

To see the version number of the XOG client
1. Bring up a command prompt.
2. Navigate to the bin directory of the XOG client and issue the command xog.
   The version number appears.

To see the version number of CA Clarity PPM
1. Log in to CA Clarity PPM.
2. Click About in the User toolbar at the top of the screen.
FIPS 140-2 Mode Setup

FIPS 140-2 is a standard that describes the U.S. federal government requirements for encrypting sensitive data. If you are using the XOG client in a FIPS 140-2 mode (-fipsenabled=true) while using an IBM JVM, additional setup is required. You must add the FIPS approved IBM JCEFIPS and IBMJSSEFIPSProvider2 providers to the provider list found in the JVM java.security file.

Provider entries in the JVM java.security file should appear similar to those shown here:

```
security.provider.1=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.2=com.ibm.crypto.provider.IBMJCE
#security.provider.3=com.ibm.jsse.IBMJSSEProvider
security.provider.3=com.ibm.jsse2.IBMJSSEProvider2
security.provider.4=com.ibm.security.jgss.IBMJGSSProvider
security.provider.5=com.ibm.security.cert.IBMCertPath
security.provider.6=com.ibm.crypto.pkcs11.provider.IBMPKCS11
security.provider.7=com.ibm.security.cmskeystore.CMSProvider
security.provider.8=com.ibm.security.jgss.mech.spnego.IBMSPNEGO
```

To add FIPS approved providers to the Java provider list

1. Open the java.security file located at `<JAVA_HOME>/jre/lib/security/java.security`.
2. Add the following IBMJCEFIPS provider entry to the beginning of the list:
   ```
   security.provider.1=com.ibm.crypto.fips.provider.IBMJCEFIPS
   ```
3. If the IBMJSSE provider entry is listed, comment it out. For example:
   ```
   #security.provider.3=com.ibm.jsse.IBMJSSEProvider
   ```
4. Add the following IBMJSSEProvider2 provider entry below the IBMJCEFIPS entry if it is not already listed:
   ```
   security.provider.<n>=com.ibm.jsse2.IBMJSSEProvider2
   ```
   Replace the `<n>` in the IBMJSSE provider entry with a number for the sequence you want the provider to be searched from the list.
5. Renumber the remaining listed entries so that they are in sequence. Verify there are no gaps in the numbers.
XOG Client Directories

The following directories are copied to your computer when you run the XOG client installer.

**bin**

This directory contains the batch files to run the XOG client and the test.properties file that can also be used to run the XOG client.

**lib**

This directory contains the libraries needed to run the XOG client.

**wsdl**

This directory contains the XOGService.wsdl file.

**xml**

This directory contains sample XML read and write files for XOG-supported objects.

**xsd**

This directory contains the XML schemas for XOG-supported objects.
Chapter 3: Running the XOG

This section contains the following topics:

- How to Run the XOG (see page 27)
- Run the XOG from the Command Line (see page 27)
- About the .properties File (see page 30)
- Run the XOG Client as a Shell (see page 32)
- How to Run the XOG Using SOAP (see page 34)

How to Run the XOG

You can run the XOG in the following ways:

- From the command line
  
  You can type in the parameters required to import and export data directly on the command line or you can store the parameters in a .properties file and call the file from the command line.

- Using the XOG client as a shell

- Using SOAP

Run the XOG from the Command Line

To run the XOG from the client using command-line parameters

1. Open a command prompt:
   
   - (Windows) From the Windows Start menu, select All Programs, CA, Clarity, CA Clarity XML Open Gateway.
   
   - (UNIX) Navigate to the XOG client home directory.

2. Type the xog command with the parameters for the operation you want to perform and press Enter.
Run the XOG from the Command Line

**Basic operations:**

- To see the command usage, issue the following command:
  
  bin\xog -?

- To read an object through the XOG, issue the following command:
  
  bin\xog -servername <host> -portnumber CA Portal -username <adminuser> -password <password> -input xml/biz_companies_read.xml

- To write output to a file (instead of displaying it in the console), issue the following command:
  
  bin\xog -servername

  By default, output is saved to the bin directory.

**Command Line Parameters (XOG Client)**

The XOG client command line uses the following parameters:

- **servername**
  
  Indicates the name of the server running CA Clarity PPM.

- **portnumber**
  
  Indicates the port number used on the server.

  **Default:** 80. The typical value for an SSL-enabled server is 443.

- **sslenabled**
  
  Indicates if the server is an SSL-enabled server.

  **Default:** False
- **output**
  Identifies the path to a file where the output of the operation should be written. Any existing file is overwritten.

- **input**
  Indicates the path to the file that contains the XOG request.

- **username**
  Indicates the username required for authentication. This user must have XOG administration access rights.

- **password**
  Indicates the user password.

- **propertyfile**
  Optional. The properties file that contains any or all of the above parameters.

- **fipsenabled**
  Indicates that the client needs to operate in a FIPS 140-2 compliant mode.
  **Default:** False
About the .properties File

You can pass command-line parameters to the XOG client from a .properties file. You can create your own .properties file from the example file provided with the XOG client's bin directory. The advantage of this method is that you can store the parameters for common XOG requests in multiple .properties files and reuse them. This saves the effort of writing out the parameters on the command line each time you want to use them.

For example, if you are using the XOG to import users, companies, resources, and content items, you might create .properties files like the ones shown here:

- users.properties
- companies.properties
- resources.properties
- content.properties

In the example .properties files listed, each file would contain different input and output property values that are appropriate for the type of data it is being used to import.

Example Properties File

The parameters needed to read project data is contained in the following .properties file example. The bolded text shows the values provided for the parameters.

```bash
# ... server host name you want to test against
servername=localhost
portnumber=80
sslenabled=false
input=./xml/prj_projects_read.xml
output=./xml/prj_projects_write.xml
username=admin
password=xogrocks
```
Create a .properties File

Make a copy of the example test.properties file located in the XOG client's bin directory and modify the copy to create a new .properties file that you can use to import or export the specific data that you need.

As you modify the file, note the following:

- Use the equal sign (=) to assign parameters in the properties file. For example, `password=admin`.
- Use the number sign (#) for comments. For example, in the figure below `#portnumber=443` is a comment that will not be read as an input value.
- The XML input file required when you run the XOG from the command line must be specified in the .properties file. The list of all read and write file examples provided in the xml folder are included. Uncomment the file you want to use for input. Be sure to comment out any files that are not being used as input.

The following figure shows the test.properties file with the default values for parameters.

```
# --- server host name you want to test against
#server hosts=localhost
#portnumber=80
#default port number for xml
#portnumber=443
#
#set to true if running against a SSL enabled server
sslenabled=false
#
#set to true if running against a SSL enabled server in FIPS 140-2 mode
fipsenabled=false
#
output=out.xml
username=admin
password=admin
#
### leave the one you want to test un-commented and comment out

#input=../xml/adminCodes_read.xml
#input=../xml/adminCodes_write.xml
#input=../xml/benefits_plan_read.xml
#input=../xml/benefits_plan_write.xml
#input=../xml/businesses_read.xml
#input=../xml/businesses_write.xml
#input=../xml/budget_plan_read.xml
#input=../xml/budget_plan_write.xml
#input=../xml/capital_scenarios_read.xml
#input=../xml/capital_scenarios_write.xml
```
Run the XOG Client as a Shell

Run the XOG Using the .properties File

To run the XOG using the .properties file
1. Modify the test.properties file or make your own .properties file and store it in the bin directory.
2. Open a command prompt:
   - Windows. From the Windows Start menu, select All Programs, CA, Clarity, CA Clarity XML Open Gateway.
   - UNIX. Go to the XOG client home directory.
3. At the XOG prompt, issue the following command:
   bin\xog -propertyfile bin/test.properties
4. View the output.

Run the XOG Client as a Shell

Typically the XOG client is used as a non-interactive command-line tool. When you are developing integrations or testing XOG requests, you may want to run the XOG client as a shell. The shell lets you log in once and execute multiple requests before logging out.

To run the XOG client as a shell
1. Open a command prompt:
   - Windows. From the Windows Start menu, select All Programs, CA, Clarity, CA Clarity XML Open Gateway.
   - UNIX. Navigate to the XOG client home directory.
2. Issue the following command:
   xog
   The shell comes up and the usage list displays.
XOG Client Shell Commands

Use the XOG client shell commands when you are developing integrations or testing XOG requests. The XOG client shell uses the following commands:

?  
  Displays the command usage screen.

login  
  Retrieves a new session ID by logging into one of the CA Clarity servers. The login command string is variable.
  
  **Example:**
  > login admin/mypassword@myserver:80

logout  
  Closes any active sessions.

output  
  Sets the path and file name where the results of the xog call will be captured.
  
  **Example:**
  > output c:\xo\xml\out.xml

call  
  Invokes a XOG request file. The file path may be absolute or relative to the working directory.
  
  **Example:**
  > call xml/biz_companies_read.xml
  > call D:/Integrations/CA Clarity/write.xml

exit  
  Logs out of active sessions and quits the shell.
How to Run the XOG Using SOAP

The XOG is a web service interface to CA Clarity. The XOG provides a SOAP API for communication with CA Clarity over the web. The XOG SOAP API is documented in the WSDL and the XSD files. Any client tool, not only the XOG client, can send and receive SOAP messages using the XOG.

The following steps describe the general process for invoking XOG directly using SOAP:

1. Call Login to establish a session.

   <SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xog="http://www.niku.com/xog">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
   <Login xmlns="http://www.niku.com/xog">
   <Username>admin</Username>
   <Password>admin</Password>
   </Login>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

   where Login Input elements are:

   Login
   The log in request main body element. Login returns a SessionID that you may use in subsequent requests.

   Username
   The name of the user doing the work.

   Password
   The password for the user.

2. Invoke the request using the SessionID.

   xmlns:xog="http://www.niku.com/xog">
   <SOAP-ENV:Header>
   <xog:Auth><xog:SessionID>[session id]</xog:SessionID></xog:Auth>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
   <obj:WriteChange xmlns:obj="http://www.niku.com/xog/Object">
   <NikuDataBus
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. Call Logout to invalidate the SessionID and close the session.

```xml
xmlns:xog="http://www.niku.com/xog">
  <SOAP-ENV:Header>
    <xog:SessionID>[session id]</xog:SessionID>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <xog:Logout/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

where Logout Input elements are:

**SessionID**

The authentication string that identifies the session to be invalidated.
Logout

The Logout request main body element.
Chapter 4: Usage Guidelines

This section contains the following topics:

- **About the Schema Files** (see page 37)
- **About the XML Read and Write Files** (see page 40)
- **Special Characters** (see page 49)
- **Date and Time Format** (see page 51)
- **Use of EQUALS, OR, BETWEEN, AFTER, and BEFORE** (see page 52)
- **Values to Pass** (see page 52)

### About the Schema Files

Schemas are templates that contain the rules for creating valid XML files that are run using the XOG. The schema definitions apply to all read and write requests and responses. You can access the schemas from the CA Clarity server or from the XOG client directories on your computer.

To find the schema definitions on the CA Clarity server, navigate to:

```
$installDir/webroot/WEB-INF/xog/xsd
```

where, `$installDir` is the customer installation directory (for example, `E:/niku/install`).

To find the schema definitions in the XOG client directories on your computer, navigate to the directory where the XOG client is installed and look in the `xsd` directory. The directory contains common schema definitions and object-specific definitions.
Schema Definitions

The following schema definitions are found in the xsd folder:

nikuxog_read.xsd (read request)

This schema definition includes:

■ nikuxog_readTypes.xsd. This schema defines the NikuDataBus request element.
■ nikuxog_readQueryTypes.xsd. This schema defines the Query request element.

Note: The nikuxog_readQueryTypes.xsd also includes the XSD files that define the read/write schemas for special stock objects.

nikuxog_<object>.xsd (read response and write request)

This schema definition applies to a read object response or a write object request.

status.xsd (write response)

This schema definition applies to all write object responses.
NikuDataBus Header Element

All read and write objects require the header element. This element is also common to all request schemas. The header defines the base version of the XOG service and the external source.

The header element has the following attributes:

version
   Required. The version of the XOG in standard XML format.
   Type: String

externalSource
   Required for Writes only. Values include:
   - NIKU
   - ORACLE-FINANCIAL
   - PEOPLESOFT
   - SAP
   - OTHER
   - OTHER-EXPENSE
   - OTHER-TIME
   - REMEDY
   Default: NIKU when reading from CA Clarity PPM
   Type: String
About the XML Read and Write Files

Attribute Information in the Schema

You can find the following attribute information in the schema:

- Sequence
- Attribute name
- Maximum field length
- Required field indicator

The following figure shows attribute information in a schema file.

About the XML Read and Write Files

Example XML read and write files for CA Clarity objects you can export and import are provided with the XOG client. These files are stored in the xml directory created when you installed the XOG client.
What is in an XML Read File

You can modify an example XML read file to create a new XML read file. Each example read file contains the necessary header information, arguments, and query filters to complete a read for the object the file represents. You can edit an example XML read file to export the information you want for an object.

The following figure shows the example XML read file for projects (prj_projects_read.xml).

- In the Header section, the action read and the objectType project indicates that this file is for exporting (reading) project data.

- The arguments indicate the types of project information you want included in the export. The default is to order the output first by name (order_by_1) and then by project ID (order_by_2). You can change the order by swapping the numbers "1" and "2" in the argument names. The default value for arguments that include data is true. Set any to false that you do not want to include in the output.

- The Query section and its filter criteria selection limit the data to only what is necessary.
How to Create an XML Write File

You can create an XML write file in the following ways:

- Create the XML write file manually
  You can use the XML write file examples in the xml folder installed when the XOG client was installed. These files are templates that can be modified to create XML files for almost any write purpose.

- Use the output of an XML read file
  The output file of an XML read file is returned in the well-formed format of an XML write file. Edit the output file to create a new XML write file. It is recommended that you use an advanced XML editor to edit this file.

Example: Create an XML Write File from the Output of an XML Read File

The following example illustrates how to create an XML write file for the project object using the output file of an XML read file.

1. Create an example project in CA Clarity that contains the information you want in the final XML write file.
   In this example, a project named project1 was created, with two resources and a task with assignments. This project information will appear in the correct XML write format in the output file.

2. Create the XML read file.
   The following code sample shows the XML read file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../xsd/nikuxog_read.xsd">
  <Header version="12.0.0.5028" action="read" objectType="project" externalSource="NIKU">
    <args name="include_tasks" value="true"/>
    <args name="include_dependencies" value="true"/>
    <args name="include_subprojects" value="true"/>
    <args name="include_resources" value="true"/>
    <args name="include_baselines" value="true"/>
    <args name="include_allocations" value="true"/>
    <args name="include_estimates" value="true"/>
    <args name="include_actuals" value="true"/>
    <args name="include_custom" value="true"/>
  </Header>
  <Query>
    <Filter name="projectID" criteria="EQUALS">project1</Filter>
  </Query>
</NikuDataBus>
```
The Header section indicates that this is a read action for the object type "project" with the list of arguments indicating the data that is to be read. The Query section indicates the name of the project for which data is to be returned.

3. Run the XOG using the read file as input.
   The output XML file is created.

4. Examine the output XML file and make any edits necessary so that you can use the file as an XML write file.

5. Save the changes.
   The XML write file is created.

The following code sample shows the output file. It is a well-formed XML write file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../xsd/nikuxog_project.xsd">
<Header action="write" externalSource="NIKU" objectType="project" version="12.0.0.5028"/>
<Projects>
<Project active="true" alignment="100" approved="true" approvedForBilling="1" asOf="2009-01-02T00:00:00" billingCurrencyCode="USD" billingType="S" calculateFinancialMetrics="true" chargeCodeExtID="Expense" clientID="Internal" clientName="Internal" closed="false" currencyISOcode="USD" description="A Test Project Description" entityCode="CORP" equipmentCostSource="Financial Cost and Rate Matrix" equipmentExchangeRateType="AVERAGE" equipmentRateSource="Financial Cost and Rate Matrix" expenseCostSource="Financial Cost and Rate Matrix" expenseExchangeRateType="AVERAGE" expenseRateSource="Financial Cost and Rate Matrix" financialStatus="O" finish="2009-12-31T17:00:00" flexibilityRisk="0" setBudgetValuesEqualToPlannedValues="true" format="0" fundingRisk="0" goalCode="IMPROVE_INFRASTRUCTURE" humanInterfaceRisk="0" implementationRisk="0" interdependenciesRisk="0"
```
About the XML Read and Write Files

laborCostSource="Financial Cost and Rate Matrix"
laborExchangeRateType="AVERAGE"
laborRateSource="Financial Cost and Rate Matrix"
lastUpdatedBy="admin" lastUpdatedDate="2009-02-11T09:42:05"
managerResourceID="paulMartin"
materialCostSource="Financial Cost and Rate Matrix"
materialExchangeRateType="AVERAGE"
materialRateSource="Financial Cost and Rate Matrix"
name="A Test Project" objectivesRisk="0"
openForTimeEntry="true" organizationalCultureRisk="0"
pageLayoutCode="projmgr.projectPageFrame"
plannedBenFinish="2010-01-01T00:00:00"
plannedBenStart="2009-12-31T00:00:00" plannedBenTotal="1000"
plannedBreakEven="2010-01-01T00:00:00"
plannedCostFinish="2010-01-01T00:00:00"
plannedCostStart="2009-01-01T00:00:00"
plannedCostTotal="1000" plannedNPV="0" plannedROI="0"
priority="10" processCode="IT" program="false" progress="0"
projectID="project1" requiredForScenarios="false"
resourceAvailabilityRisk="0" sponsorshipRisk="0"
stageCode="CSK_INITIATION" start="2009-01-01T00:00:00"
status="1" statusComment="Status Comment Text"
statusIndicator="1" supportabilityRisk="0"
syncInvestmentAndBudgetDates="true" technicalRisk="0"
template="false" trackMode="2">
<Resources>
    <Resource availFrom="2009-01-01T08:00:00"
        availTo="2009-12-31T17:00:00" bookingStatus="5"
        defaultAllocation="1" isProjectManager="false"
        lastUpdatedBy="admin"
        lastUpdatedDate="2009-02-11T09:39:40"
        openForTimeEntry="true"
        projectRoleID="csk.Architect" resourceID="artKatect">

    </Resource>
</Resources>
The topics that follow describe how to perform the tasks most routine to this interface.

```xml
<Task finish="2009-12-31T17:00:00" internalTaskID="5000578" key="false"
    lastUpdatedBy="admin"
    lastUpdatedDate="2009-02-11T09:40:11"
    lockedForScheduling="false" milestone="false"
    name="A Test Project" orderID="1" outlineLevel="1"
    percComp="0" start="2009-01-01T08:00:00" status="0"
    summary="false" taskID="~rmw" topDownPercent="0">
    <Baselines/>
    <Assignments>
        <TaskLabor actualWork="0" baselineWork="0"
            estPattern="3" finish="2009-12-31T17:00:00"
            lastUpdatedBy="admin"
            lastUpdatedDate="2009-02-11T09:41:41"
            remainingWork="2088" resourceID="artKatect"
            roleID="csk.Architect"
            start="2009-01-01T08:00:00" unpostedActuals="0">
        </TaskLabor>
    </Assignments>
    <EstCurve>
        <Segment finish="2010-01-01T00:00:00" />
    </EstCurve>
</Task>
```

```xml
</Resources>
```

The topics that follow describe how to perform the tasks most routine to this interface.

```xml
<Resource availFrom="2009-01-01T08:00:00"
    availTo="2009-12-31T17:00:00" bookingStatus="5"
    defaultAllocation="1" isProjectManager="true"
    lastUpdatedBy="admin"
    lastUpdatedDate="2009-02-11T09:40:11"
    openForTimeEntry="true"
    projectRoleID="csk.Project Manager"
    resourceID="paulMartin">
    <Baselines/>
    <AllocCurve/>
    <CustomInformation>
        <ColumnValue
            name="partition_code">NIKU.ROOT</ColumnValue>
    </CustomInformation>
    <SkillAssocs/>
</Resource>

```xml
</Resource>
</Resources>
```
<TaskLabor actualWork="0" baselineWork="0"
estPattern="3" finish="2009-12-31T17:00:00"
lastUpdatedBy="admin"
lastUpdatedDate="2009-02-11T09:40:11"
remainingWork="2088" resourceID="paulMartin"
roleID="csk.Project Manager"
start="2009-01-01T00:00:00" unpostedActuals="0">
<Baselines/>
<EstCurve>
  <Segment finish="2010-01-01T00:00:00"
    start="2009-01-01T00:00:00"
    sum="2088.0000"/>
</EstCurve>
<ActCurve/>
<CustomInformation>
  <ColumnValue
    name="partition_code">NIKU.ROOT</ColumnValue>
</CustomInformation>
</TaskLabor>
</Assignments>
<estimateRules/>
<EstCurve>
  <Segment finish="2010-01-01T00:00:00" start="2009-01-01T00:00:00" sum="2088.0000"/>
</EstCurve>
<ActCurve/>
<CustomInformation>
  <ColumnValue name="partition_code">NIKU.ROOT</ColumnValue>
</CustomInformation>
</EstCurve>
<CustomInformation>
  <ColumnValue name="partition_code">NIKU.ROOT</ColumnValue>
</CustomInformation>
</TaskLabor>
<Assignments>
<CustomInformation>
  <ColumnValue name="partition_code">NIKU.ROOT</ColumnValue>
</CustomInformation>
</Assignments>
</Tasks>
<Dependencies/>
<Subprojects/>
<Allocations/>
<scenarioDependencies/>
<InvestmentAssociations>
  <Allocations/>
  <Hierarchies/>
</InvestmentAssociations>
<CustomInformation>
  <ColumnValue name="obj_align_factor1">50</ColumnValue>
  <ColumnValue name="obj_align_factor2">50</ColumnValue>
  <ColumnValue name="obj_align_factor3">50</ColumnValue>
  <ColumnValue name="obj_align_factor4">50</ColumnValue>
</CustomInformation>
Special Characters

You must escape special characters in XOG requests to help ensure a successful XOG read or write request. You can escape special characters or use CDATA.

Use Escape Rules

You can use one of the following escape rules to escape special characters in the XML file.

<table>
<thead>
<tr>
<th>Special Character</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp; (Ampersand)</td>
<td>&amp;</td>
</tr>
<tr>
<td>' (Apostrophe)</td>
<td>''</td>
</tr>
<tr>
<td>&gt; (Greater-than)</td>
<td>&gt;</td>
</tr>
</tbody>
</table>

Apostrophes must be double-escaped as shown.
<table>
<thead>
<tr>
<th>Special Character</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; (Less-than)</td>
<td>&lt;</td>
</tr>
<tr>
<td>&quot; (Quotes)</td>
<td>&quot;</td>
</tr>
</tbody>
</table>

**Escape Example**

The following example shows how to handle the term \$E1&P2\$ in XML text by escaping the term:

```xml
<ColumnValue name="abn_vendor_names">E1&amp;P2</ColumnValue>
```

**Use CDATA**

You can use CDATA instead of escaping special characters. CDATA is a section of element content in XML that is marked so that it is interpreted only as character data, not markup data.

To start a CDATA section, use:

```xml
<![CDATA[
```

To end a CDATA section, use:

```xml
]]>
```

**CDATA Example**

The following example shows how to handle the term \$E1&P2\$ in XML text using CDATA.

```xml
<ColumnValue name="abn_vendor_names"><![CDATA[E1&P2]]></ColumnValue>
```
Date and Time Format

You must format date and time strings in the following standard format for the XOG:

- **Date format:** YYYY-MM-DD
- **Time format:** HH24MMSS

Note the following:

- **Date and time values for custom objects**
  
  The date and time value of a date attribute is stored in Greenwich Mean Time (GMT). For a custom object, you must offset the date and time value being stored from the locale you are in to GMT. For example, if the desired date and time is November 20th, 2008, 15:15 (3:15PM) in Tokyo, the date and time value in the XOG write file needs to be formatted and adjusted to GMT time (which in Tokyo is +9 hours). So the resulting offset-formatted value to be entered in the XOG import file would be 2008-11-21T00:15:00.

- **Timestamps for task finish dates**

  Include a timestamp for task finish dates in the prj_projects_write.xml file. If you do not, the time defaults to 00:00:00. The effect of the default is that in Portfolio portlets, the finish dates with the default timestamp applied display a day later. For example, 2009-01-01 shows in the Portfolio portlets, when the actual finish date is 2008-12-31. To avoid the addition of an extra day to a finish date, use 17:00:00 as the timestamp when one is not provided.
Use of EQUALS, OR, BETWEEN, AFTER, and BEFORE

Filtering in a XOG read request requires criteria values. Possible criteria values include:

- EQUALS
- OR
- BEFORE
- AFTER
- BETWEEN

Examples:

```xml
<Filter name="projectID" criteria="EQUALS">test</Filter>
<Filter name="projectID" criteria="OR">project1,project2</Filter>
<Filter name="projectID" criteria="AFTER">A</Filter>
<Filter name="projectID" criteria="BEFORE">Z</Filter>
<Filter name="start" criteria="BETWEEN">2007-01-07,2009-01-15</Filter>
```

Important! No spaces should be used around comma-separated entries for OR and BETWEEN filters.

Values to Pass

The following table shows the values expected by the XOG in specific cases.

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Value Type Passed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lookup</td>
<td>lookup_code, lookup_enum, or lookup ID. The type passed depends on the configuration of the lookup.</td>
</tr>
<tr>
<td>Custom Boolean field</td>
<td>1 or 0</td>
</tr>
</tbody>
</table>
This section contains the following topics:

- **Object API** (see page 53)
- **InvokeAction API** (see page 68)
- **Query API** (see page 71)

## Object API

The structure for each object is defined in its corresponding schema (XSD). There is one general schema definition for all read object requests; however, the schema definition for each read object response and each write object request varies based on the object that is accessed. This is because the structure for each object is defined in its own corresponding schema.

See Appendix A, XOG Object Reference for a complete list of available XOG objects.

## XOG Object Types

The Object API includes the following group of XOG objects:

**Standard stock objects**

Refers to actual objects in the XOG. You can communicate with these standard stock objects by reading and writing data using a valid interface.

**Custom objects and add-ins**

These objects have unique schemas that differ from the standard stock object schema. You must first create custom objects in Studio before you can invoke them through the XOG. Add-in schemas allow you to read and write Studio components. For example, objects, pages, and portlets.

For more information, see the *Studio Developer’s Guide*. 
**ActionObject Element**

All read and write XOG objects use an `<ActionObject>` element to define the action to be taken and the object to take it on. An action can be either a read or write. An object can be any XOG object that is available under the Objects API category such as department and companies.

For example, the `<ReadProject>` element indicates read as the action to take, and Project is the XOG object against which the action is taken. The `<ActionObject>` element is the parent element that wraps around the NikuDataBus header attributes.

The following example shows the structure used for requesting a read action on the Project object:

```xml
<obj:ReadProject xmlns:obj="http://www.niku.com/xog/Object">
  <NikuDataBus>
    <Header/>
    <Query/>
  </NikuDataBus>
</obj:ReadProject>
```

The namespace http://www.niku.com/xog/Object must be localized in your request file to the `<ActionObject>` element. This localization is accomplished with the obj prefix. Failure to include this prefix results in an error during the processing of the request.
Read Standard Stock Objects

Example XML Read Request

Read object requests are used to extract specific object records from CA Clarity. The read object request services reference the nikuxog_read.xsd schema (shown in bold) in the following example.

```xml
<obj:ReadUser xmlns:obj="http://www.niku.com/xog/Object">
  <NikuDataBus
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../xsd/nikuxog_read.xsd">
    <Header version="12.0.0.5028" externalSource="NIKU"/>
    <Query>
      <Filter name="userName" criteria="EQUALS">admin</Filter>
    </Query>
  </NikuDataBus>
</obj:ReadUser>
```

Example XML Read Response

In the following example, the nikuxog_user.xsd defines the NikuDataBus read response element.

```xml
<ReadUserResponse xmlns="http://www.niku.com/xog/Object">
  <NikuDataBus
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../xsd/nikuxog_user.xsd">
    <Header externalSource="NIKU" version="12.0.0.5028"/>
    <Users>
      <User externalId=" " userLanguage="English"
        userLocale="en_US" userName="admin"
        userStatus="ACTIVE" userTimezone="Europe/London"
        userType="INTERNAL">
        <PersonalInformation emailAddress="test@ca.com"
          firstName="ca" lastName="Administrator"/>
        <Resource resourceId="admin"/>
        <Company/>
        <General addedBy="admin" addedDate="2005-03-12"/>
        <OBSAssocs>
          <OBSAssoc id="loc" name="Location" unitPath="/USA"/>
        </OBSAssocs>
        <Groups/>
        <GlobalRights/>
        <InstanceRights/>
        <InstanceOBSRights/>
      </User>
    </Users>
  </NikuDataBus>
</ReadUserResponse>
```
</User>
</Users>
<XOGOutput>
  <Status state="SUCCESS"/>
  <Statistics failureRecords="0" insertedRecords="0"
    totalNumberOfRecords="1" updatedRecords="0"/>
  </Records>'
</XOGOutput>
</NikuDataBus>
</ReadUserResponse>

Write Standard Stock Objects

Write object requests are used to insert and update records into another system.

例 XML Write Request

In XML write requests, the unique record identifier varies based on the object. In the following example, the identifier is the userName attribute. This example is an update, and it includes only a subset of the information exported in the read request example. The nikuxog_user.xsd defines the NikuDataBus write request element.

例 XML Write Request

例 XML Write Request

例 XML Write Request

例 XML Write Request

例 XML Write Request

例 XML Write Request

例 XML Write Request

例 XML Write Request

例 XML Write Request

例 XML Write Request

例 XML Write Request
Example XML Write Response

The following is the result from the example write request.

```xml
<WriteUserResponse xmlns="http://www.niku.com/xog/Object">
  <XOGOutput xsi:noNamespaceSchemaLocation="/../xsd/status.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Object type="user"/>
    <Status state="SUCCESS"/>
    <Statistics failureRecords="0" insertedRecords="0"
      totalNumberOfRecords="1" updatedRecords="1"/>
    <Records>
      <Record>
        <KeyInformation>
          <column name="ALL">ALL RECORDS</column>
        </KeyInformation>
        <ErrorInformation>
          <Severity>WARNING</Severity>
          <Description>New Users Password will be Defaulted to Value ca2000</Description>
        </ErrorInformation>
      </Record>
    </Records>
  </XOGOutput>
</WriteUserResponse>
```
Partitioning and Standard Stock Objects

The read object response services provide partition view information for each object instance. Similarly, you can write a partition view to each write object instance request.

If you do not specify a partition view in your write request, all the instances you create are automatically assigned to the default system partition value NIKU.ROOT.

To specify a new partition, replace NIKU.ROOT with your actual partition code. Before you can specify a partition view, you must create a partition model and associate it with your objects in Studio.

For more information, see the Studio Developer’s Guide.

Example Partition XML

The following example shows how partition information is specified for each object instance (that is for each "Project") using the <CustomInformation> element.

```xml
<obj:WriteProject xmlns:obj="http://www.niku.com/xog/Object">
  <NikuDataBus xsi:noNamespaceSchemaLocation="../xsd/nikuxog_user.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Header externalSource="NIKU" version="8.0"/>
    <Projects>
      <Project>
        ...
        ...
        ...
        ...
        ...
        <CustomInformation>
          <ColumnValue name="partition_code">NIKU.ROOT</ColumnValue>
        </CustomInformation>
      </Project>
      ...
      ...
      ...
    </Projects>
  </NikuDataBus>
</obj:WriteProject>
```
Custom Object Instances

The CustomObjectInstances service is an entry point to enable XOG communication with instances of custom objects. Instances represent data held within custom objects, not the definition of the objects.

For more information, see the Studio Developer’s Guide.

Read CustomObjectInstances

A CustomObjectInstances read request requires the namespace niku_xog_read.xsd and then the <CustomObjectInstanceQuery> element.

The CustomObjectInstanceQuery element allows you to filter on instances of one or more custom objects using the following filter attributes:

objectCode

Refers to the custom object ID as defined in Studio.

instanceCode

Refers to the custom object instance ID as defined in Studio.

For more information, see the Studio Developer’s Guide.

Example XML Read Request

The following XML shows an example read CustomObjectInstance request and how it uses the <CustomObjectInstanceQuery> element.

```xml
<obj:ReadCustomObjectInstance xmlns:obj="http://www.niku.com/xog/Object">
  <NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../xsd/nikuxog_read.xsd">
    <Header version="12.0.0.5028" externalSource="NIKU"/>
    <CustomObjectInstanceQuery>
      <Filter name="objectCode" criteria="EQUALS">training_modules</Filter>
      <Filter name="instanceCode" criteria="EQUALS">Business Ethics</Filter>
    </CustomObjectInstanceQuery>
  </NikuDataBus>
</obj:ReadCustomObjectInstance>
```
Write CustomObjectInstances

The write CustomObjectInstances request services are defined by the nikuxog_customObjectInstance.xsd schema.

Example Write XML

The following example shows an XML write request:

```xml
<obj:WriteCustomObjectInstance xmlns:obj="http://www.niku.com/xog/Object">
  <NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="..//xsd/nikuxog_customObjectInstance.xsd">
    <Header externalSource="NIKU" version="12.0.0.5028"/>
    <customObjectInstances objectCode="movies">
      <instance instanceCode="Star Wars" objectCode="movies11">
        <CustomInformation>
          <ColumnValue name="category">Science Fiction</ColumnValue>
          <ColumnValue name="code">Star Wars</ColumnValue>
          <ColumnValue name="cost">20000000</ColumnValue>
          <ColumnValue name="cost_currency">USD</ColumnValue>
          <ColumnValue name="name">Star Wars</ColumnValue>
          <ColumnValue name="partition_code">US</ColumnValue>
          <ColumnValue name="us_rating">PG-13</ColumnValue>
        </CustomInformation>
      </instance>
    </customObjectInstances>
  </NikuDataBus>
</obj:WriteCustomObjectInstance>
```

Partitioning

Like standard stock objects, the read CustomObjectInstances response service provides partition view information for each custom object instance. You can write a partition view to each write CustomObjectInstances instance. In the previous XML write request example, the partition view of US is specified for the Star Wars movie instance definition.

ContentPack Service

The ContentPack service is an entry point to enable XOG communication with Studio components, such as objects, object views, NSQL queries, portlets, process definitions, report definitions, lookups, and portlet pages.

For more information, see the Studio Developer's Guide.
Read Content Pack Objects

Like standard stock objects, the read ContentPack request service is also defined by the nikuxog_read.xsd schema as shown in the following example. However, for the ContentPack service, this schema is unique because it includes the <ViewQuery> element. This element allows you to read Studio components.

Example XML Read Request

The following example shows the ViewQuery for reading the property and list components of the custom object myObject_v1.

```xml
<obj:ReadContentPack xmlns:obj="http://www.niku.com/xog/Object">
  <NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../xsd/nikuxog_read.xsd">
    <Header version="12.0.0.5028" externalSource="NIKU"/>
    <ViewQuery>
      <Filter name="code" criteria="EQUALS">property</Filter>
      <Filter name="object_code" criteria="EQUALS">myObject_v1</Filter>
    </ViewQuery>
    <ViewQuery>
      <Filter name="code" criteria="EQUALS">list</Filter>
      <Filter name="object_code" criteria="EQUALS">myObject_v1</Filter>
    </ViewQuery>
  </NikuDataBus>
</obj:ReadContentPack>
```
Read Content Pack Objects with Partitioning

You can filter on partition views from a ContentPack read request by including the <ViewQuery> filter condition attribute.

To read specific partitioned views using the ContentPack XOG, you must explicitly request these partitions in a ViewQuery by including the partition_code.

In the file, specify the following:

object_code

Indicates the CA Clarity identifier for the object.

partition_code

Indicates the CA Clarity identifier for the partition. If the partition_code is not specified, views for all partitions are exported.

Example XML Read Request with Partitioning

The following read ContentPack request specifies the ABC partition for the custom object myObject_v1 as shown in bold.

```xml
<obj:ReadContentPack xmlns:obj="http://www.niku.com/xog/Object">
  <NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../xsd/nikuxog_read.xsd">
    <Header version="12.0.0.5028" externalSource="NIKU">
      <!-- the contentType is used to determine which filter goes where -->
      <args contentType="job_definition" name="order_by_1" value="code"/>
      <args contentType="menu" name="order_by_1" value="code"/>
      <args contentType="view" name="order_by_1" value="code"/>
      <args contentType="process" name="order_by_1" value="code"/>
      <args contentType="object" name="order_by_1" value="code"/>
    </Header>
    <ViewQuery>
      <Filter name="code" criteria="EQUALS">property</Filter>
      <Filter name="object_code" criteria="EQUALS">myObject_v1</Filter>
      <Filter name="partition_code" criteria="EQUALS">ABC</Filter>
    </ViewQuery>
    <ViewQuery>
      <Filter name="code" criteria="EQUALS">list</Filter>
      <Filter name="object_code" criteria="EQUALS">myObject_v1</Filter>
      <Filter name="partition_code" criteria="EQUALS">ABC</Filter>
    </ViewQuery>
    <ObjectQuery>
      <Filter name="object_code" criteria="EQUALS">myObject_v1</Filter>
    </ObjectQuery>
  </NikuDataBus>
</obj:ReadContentPack>
```
Export Content Types Without Dependencies

You can use two arguments, singleContentType and no_dependencies, to export individual content types and to limit the amount of data exported for each type. These arguments should be used only for small, incremental updates to content.

Important! Use care when applying these arguments. The user performing the export must have a thorough understanding of content data on the source and target systems.

singleContentType

Exports a specific content type. You must specify the singleContentType argument in the XML read file. The format for the argument in the XML read file is as follows:

<args name="singleContentType" value="content type">

where content type can be any of the following supported content types:

- Job Definitions
- Lookups
- Menu Manager
- Objects
- Portlet Pages
- Portlets
- Processes
- Queries

No_dependencies

Limits the amount of data exported for a specific content type. The format for the argument in the XML read file is as follows:

<args name="no_dependencies" value="true/false"/>

where true exports incremental changes for a content type without dependencies. By default, the no_dependencies flag is false, which means any dependencies that exist for the value specified in the singleContentType argument are exported if the value is not specified.
The following table shows the content types that are exported when the `no_dependencies` argument is set to true.

<table>
<thead>
<tr>
<th>Content Type</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>job Definition</td>
<td>active, executable, isHidden, jobDefinitionCode, jobType, logEnabled, outputEnabled, runConcurrent, source, description, name, attributes related to parameter like code, dataType, defaultValue, order, readOnly, required, widgetType, attributes related to OBSAssocs like completed, Security</td>
</tr>
<tr>
<td>Lookup</td>
<td>code, hiddenAttributeName, sortStype, source, status, description, name, sortorder, attributes related to partition</td>
</tr>
<tr>
<td>Menu</td>
<td>code, source, description, name, link, section</td>
</tr>
<tr>
<td>Object</td>
<td>code, source, description, name, displayMapping, parentObjectCode</td>
</tr>
<tr>
<td>Page (portlet page)</td>
<td>tabbedPage, code, customizable, layout, linkable, objectTypes, personalizable, source, space, template, description, name, tab, OBSAssocs, Parameter, dataref, dataSource, paramCode</td>
</tr>
<tr>
<td>Portlet</td>
<td>allowConfigure, allowConfigureLabel, category, code, colorItem, dataProviderId, dataProviderPartitionId, dataProviderType, datapointLabels, firstSliceAngle, forceFilter, link, mouseoverLabels, objectType, seriesDimension, showLegend, showTitle, size, source, type, name, decimalPlaces <strong>Note</strong>: Attributes can vary depending on type of portlet.</td>
</tr>
<tr>
<td>Process</td>
<td>code, endStep, allowOneRunningInstance, source, startOption, startStep, description, name, rightCode, username, manualStart, objectType, partitionCode, partitionModeCode, type, isMilestone, sequenceNo, attributes related to Notifications, Operations, TransactionRestrictions</td>
</tr>
<tr>
<td>Query</td>
<td>category, code, description, name, attributes related to nsq1 like dbId, dbVendor, attributes for link like action, code</td>
</tr>
</tbody>
</table>
Example: XML Read File for Exporting a Portlet Without Dependencies

The following example shows an XML read file where portlet data without any dependency data is requested. The arguments are shown in bold.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../xsd/nikuxog_read.xsd">
  <Header version="12.0.0.5028" action="read" objectType="contentPack" externalSource="NIKU">
    <!-- Provide following argument with singlecontenttype to retrieve only portlet. -->
    <args name="singleContentType" value="portlet" />
    <!-- May specify following OPTIONAL argument no_dependencies to exclude dependent content. -->
    <args name="no_dependencies" value="true" />
  </Header>
  <PortletQuery>
    <Filter name="code" criteria="EQUALS">balance</Filter>
  </PortletQuery>
</NikuDataBus>
```

Export Portlet Data from Studio to an XML File

To facilitate the export of portlets, you can export this content type from Studio. The Portlets list page has an Export button that lets you export basic data on the portlets as individual XML files. The output XML files, which are packaged into a zip file, contain basic information with no dependencies.
Autonumbering and Custom Attributes

The flag `overrideAutoNumbering` is an argument defined in the XOG header that determines whether source XOG content overrides autonumbering in the target content. The flag is available for the custom attributes of custom objects.

The following rules apply:

- If the flag is set to TRUE, XOG content from the source is applied to the target.
- If the flag is set to FALSE, the autonumbering scheme defined on the target is applied.
- The flag is specified in the XOG import file.

By default, `OverrideAutoNumbering=TRUE`.

The following example shows the `overrideAutoNumbering` argument in the header:

```xml
<Header action="write" externalSource="NIKU" objectType="customObjectInstance" version="13.0.0">
  <args name="overrideAutoNumbering" value="false"/>
</Header>
```

Import and Export Custom Fiscal Time-Varying Attributes

You can export or import custom fiscal time-varying attributes as part of the import or export of a standard or custom subobject with which the attributes are associated.

When a subobject is exported, the XOG export file includes the following elements:

- Custom attributes and modified views for the subobject
- Custom attributes and modified views for the master object
- Referenced lookups

If a subobject includes custom fiscal time-varying attributes, the master object export file includes the following additional elements:

- Fiscal entity
- Fiscal time period
- Department OBS associations
Import and Export UI Themes

You can import and export UI themes through the XML Open Gateway. The XOG client provides sample UI theme XML files. The import file for UI themes is `cmn_ui_themes_write.xml`.

The `default` attribute on the UITheme element determines whether a UI theme being imported is the default theme for the system. If the `default` attribute is set to true, the theme becomes the system default theme. If you are importing multiple UI themes in one import file, only one UI theme is expected to have the `default` attribute set to true. If multiple themes in a single import have the `default` attribute set to true, the last theme processed with the attribute set to true becomes the default theme.

Sample Import File `cmn_UI_themes_write.xml`

The following section of the `cmn_ui_themes_write.xml` shows the `default` attribute set to false. In this case, the UI theme being imported is not the default CA Clarity UI theme.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright (c) 2011, CA Inc. All rights reserved. -->
<NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../xsd/nikuxog_uitheme.xsd">
  <Header action="write" externalSource="NIKU" objectType="uitheme" version="13.0"/>
  <UIThemes>
    <UITheme active="true" default="false" id="sample_ui_theme">
      <nls description="A sample UI Theme" languageCode="en" name="Sample UI Theme"/>
      <css><![CDATA[...
```
Include New and Delete Buttons

When you are importing a new custom object, the New and Delete buttons are deselected by default. This means it is not possible to immediately create a new instance of the object. To make the New and Delete buttons available immediately for the new object, include the following two lines in the script:

```
<action code="odf.@objectCode@Create"/>
<action code="odf.deleteObjectInstancesConfirm"/>
```

Example:

```
<list>
  ....
  <action code="odf.testCreate"/>
  <action code="odf.deleteObjectInstancesConfirm"/>
  ....
</list>
```

About Passing XDM Custom Fields

XDM configuration changes are automatically handled by the Object API. The following rules apply:

- For the Name and Values fields, use those defined in the customFieldsMetadata.xml file.
- For lookups, pass the lookup code and dates (in YYYY-MM-DD format).
- For checkbox fields, pass 1 or 0.

Example

```
<CustomInformation>
  <ColumnValue name="CEO_NAME">ceo2</ColumnValue>
  <ColumnValue name="DEFAULTWEBSITE">http://www1</ColumnValue>
  <ColumnValue name="NUM_OF_EMPLOYEES">100</ColumnValue>
  <ColumnValue name="OPPORTUNITY">1</ColumnValue>
</CustomInformation>
```

InvokeAction API

The InvokeAction API is a general-purpose area for administrative actions that fall outside the categories of data import and export. There is no corresponding schema (XSD) for this API and accordingly there is no XSD validation.
InvokeAction API Root Elements

The following are the root elements of the InvokeAction API:

FlushCache

This action is used to flush a cache within a running Application.

Process

This action is used to schedule integration processes or initiate them in realtime.

FlushCache Elements

The following describes the FlushCache root elements:

group

Optional. Identifies the cache group to be flushed.

Type: String

id

Optional. Identifies the cache ID to be flushed.

Type: String

Example: XML Flush Cache

In the following example, the xmlns= attribute is using the InvokeAction API. There is no reference to a schema or XSD.

<FlushCache xmlns="http://www.niku.com/xog/InvokeAction">
  <group>Resources</group>
  <id>ConfigurationProperties</id>
</FlushCache>
Process Elements

The following describes the Process root elements:

**code**
Required. Identifies the process ID.

**Type:** String

**request**
Optional. Identifies the root element of the process input document.

**Type:** Any

**Example: Process Request**

In the following sample, remedy_writeIncident is the process ID required to invoke the process action. Note that a process may or may not include a request. An action can be invoked by specifying the process ID. The following example includes a request.

```xml
<Process xmlns="http://www.niku.com/xog/InvokeAction">
  <code>remedy_writeIncident</code>
  <request>
    <incidents>
      <incident assignedTo="jstewart"
        categoryCode="telcom"
        estimatedEffort="240"
        estimatedEffortUnit="MINUTES"
        externalId="tc421"
        impactCode="High"
        incidentCode="RMD-TC421"
        priorityCode="Medium"
        reportedBy="rcordry"
        resolutionDate="2005-03-03T12:30:00"
        sourceCode="REMEDY"
        startDate="2005-03-01T08:00:00"
        statusCode="Closed"
        subject="Phone system down"
        typeCode="incident"
        urgencyCode="High">
      
      <description>
        Tried making call, no dial tone.
      </description>
      
      <notes/>
      
      <efforts enterOnce="true">
        <effort quantity="3.5" quantityUnit="HOURS"
          resourceCode="jstewart"
          transactionDate="2005-03-03"/>
      </efforts>
      
      <contacts/>
    </incident>
  </incidents>
</Process>
```
Query API

It is often not sufficient to read data only as predefined objects. For example, full object instance data including dependencies may be too much detail. Or, you may need data from multiple objects or from database tables that have no defined objects.

You can use the Query API to execute Studio NSQL-based queries from the XOG. The Studio Query is referenced by its code. The response is formatted as record elements.

**Note:** The Query API requires a valid license for Studio.

**Example: Studio Query**

code: sample.getresources

**NSQL:**

```sql
SELECT
@SELECT:DIM:USER_DEF:IMPLIED:RESOURCE:R.FULL_NAME:RSRC@,
@SELECT:DIM_PROP:USER_DEF:IMPLIED:RESOURCE:MR.FULL_NAME:MGR@,
@SELECT:METRIC:USER_DEF:IMPLIED:COUNT(*):PROJECT_COUNT:AGG@
FROM
  SRM_PROJECTS P,
  SRM_RESOURCES R,
  SRM_RESOURCES MR,
  CMN_SEC_USERS U
WHERE    P.CREATED_BY = U.ID
AND      U.ID = R.USER_ID
AND      R.MANAGER_ID = MR.USER_ID
AND      @FILTER@
GROUP BY R.FULL_NAME,
         MR.FULL_NAME
HAVING   @HAVING_FILTER@
```

**Example: XML Query**

```xml
<Query xmlns="http://www.niku.com/xog/Query">
  <Code>sample.getresources</Code>
</Query>
```
Example: Result

```xml
<QueryResult xmlns="http://www.niku.com/xog/Query"
  <Code>sample.getresources</Code>
  <Records>
    <Record>
      <rsrc>Administrator, Niku</rsrc>
      <project_count>178</project_count>
      <manager>Administrator, Niku</manager>
    </Record>
  </Records>
</QueryResult>
```

Query API Root Elements

The following describes the Query API root elements:

**Code**

Required. Identifies the NSQL unique identifier defined in Studio.

Type: String

**Filter**

Optional. Identifies the NSQL filter columns defined in Studio.

Type: User-defined

The Query Filter

The WSDL for queries defines filter parameters in addition to the code identifier. This allows ad-hoc queries based on a Studio Query using the XOG. For every column selected in the query, you are given multiple filter possibilities.
**Exact Match**

To filter on a specific value for a column, use the column name directly and pass the value in which you are interested.

The following example retrieves a single row for resource CorpApp Administrator. Any deviation in the rsrctype value of 'Administrator, CorpApp' returns nothing.

**Example**

```xml
<Query xmlns="http://www.niku.com/xog/Query">
  <Code>sample.getresources</Code>
  <Filter>
    <rsrc>Administrator, CorpApp</rsrc>
  </Filter>
</Query>
```

**Wildcard Query Example**

The wildcard filter behaves like any grid filter field. It automatically appends a wildcard asterisk (*) to the end of a value. You can also insert your own asterisk anywhere in the filter string including at the beginning. The latter is not recommended when filtering very large data sets, as performance is severely degraded.

The wildcard filter is available only on columns of type String.

**From the sample.getresources Example**

```xml
<Query xmlns="http://www.niku.com/xog/Query">
  <Code>sample.getresources</Code>
  <Filter>
    <rsrc_wildcard>Admin</rsrc_wildcard>
  </Filter>
</Query>
```

**Another Example**

```xml
<Query xmlns="http://www.niku.com/xog/Query">
  <Code>sample.getresources</Code>
  <Filter>
    <rsrc_wildcard>Admin*CorpApp</rsrc_wildcard>
  </Filter>
</Query>
```
Capture Bounded and Unbounded Ranges

The from and to filters perform a "greater than or equal to" and "less than or equal to" operation on a given value. Use these filters to capture a bounded range or separately for an unbounded range.

**Example: From Filter**

The following unbounded example returns all records with project_count greater than or equal to 1:

```xml
<Query xmlns="http://www.niku.com/xog/Query">
  <Code>sample.getresources</Code>
  <Filter>
    <project_count_from>1</project_count_from>
  </Filter>
</Query>
```

**Example: To Filter**

The following bounded example returns all records with rsrc values that start with the letters A through E:

```xml
<Query xmlns="http://www.niku.com/xog/Query">
  <Code>sample.getresources</Code>
  <Filter>
    <rsrc_from>A</rsrc_from>
    <rsrc_to>E</rsrc_to>
  </Filter>
</Query>
```
Example: Exporting Query Results to a Tab-Delimited Text File

The following example uses GEL to execute the NSQL query "xog_query_test", which is the default NSQL query. The example shows how to retrieve the query results and export them to a tab-delimited text file.

For more information on using GEL with the XOG, see the chapter named "GEL Scripting (see page 77)."

Example

```xml
gel:script
  xmlns:core="jelly:core"
  xmlns:xog="http://www.niku.com/xog"
  xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary"
  xmlns:soap="jelly:com.niku.union.gel.SOAPTagLibrary"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap-envelope/"
  xmlns:f="jelly:com.niku.union.gel.FileTagLibrary"
  xmlns:nikuq="http://www.niku.com/xog/Query"
  xmlns:util="jelly:util">
  <!- Construct the Query API request for the NSQL query "xog_query_test" -->
  <gel:parse var="xoginput">
    <Query xmlns="http://www.niku.com/xog/Query">
      <Code>xog_query_test</Code>
    </Query>
  </gel:parse>

  <soap:invoke endpoint="http://localhost/niku/xog" var="xogresponse">
    <soap:message>
      <soapenv:Envelope>
        <soapenv:Header>
          <Auth>
            <Username>admin</Username>
            <Password>niku2000</Password>
          </Auth>
        </soapenv:Header>
        <soapenv:Body>
          <gel:include select="$xoginput"/>
        </soapenv:Body>
      </soapenv:Envelope>
    </soap:message>
  </soap:invoke>

  <!- Extract the sessionID so we may logout later -->
  <gel:set asString="true" select="$xogresponse/soapenv:Envelope/soapenv:Body/xog:SessionID/text()" var="sessionID"/>
  <gel:out>SessionID = ${sessionID}</gel:out>
```
<!-- Extract the records -->
<gel:set
select="$xogresponse/soapenv:Envelope/soapenv:Body/nikuq:QueryResult/nikuq:Records" var="records"/>
<gel:set asString="true" select="$records" var="recordstext"/>
<gel:out>${recordstext}</gel:out>

<!-- Create a tab-delimited file from the results -->
<f:writeFile fileName="projectData.txt" delimiter="\t" embedded="true">
<gel:forEach select="$records//nikuq:Record" var="xog_record">
  <f:line>
    <gel:forEach select="$xog_record/*" var="xog_data">
      <gel:set var="xog_data" select="$xog_data/text()" asString="true"/>
      <f:column value="${xog_data}"/>
    </gel:forEach>
  </f:line>
</gel:forEach>
</f:writeFile>

<!-- Now log out -->
<soap:invoke endpoint="http://localhost/niku/xog" var="logout">
  <soap:message>
    <soapenv:Envelope>
      <soapenv:Header>
        <Auth>
          <xog:SessionID>${sessionID}</xog:SessionID>
        </Auth>
      </soapenv:Header>
      <xog:Logout/>
    </soapenv:Body>
  </soapenv:Envelope>
</soap:invoke>
</gel:script>
Chapter 6: GEL Scripting

This section contains the following topics:

GEL Overview (see page 77)
GEL Setup (see page 78)
GEL Script Validation and Execution (see page 78)
GEL Basics (see page 78)
Set GEL Tag Restrictions (see page 84)
Examples for Running the XOG (see page 85)
Database Operations (see page 89)
File Operations (see page 91)
Integration Processes (see page 96)

GEL Overview

Important! Before you use GEL, read the Customization Policy. See your CA account representative.

GEL (Generic Execution Language) is a tool you can use to turn XML into executable code. It is based on Jelly, a jakarta.apache.org Commons project. It has been extended and embedded into CA Clarity PPM to enable custom logic to solve business problems. GEL is the basis for the enterprise application integration framework within CA Clarity PPM.

GEL also provides a collection of standard integrations that provide connectors to enterprise applications such as Remedy® Help Desk.

With GEL you can invoke and process a variety of data sources:

Web services

GEL can read or write to any SOAP-based web service. This includes the XOG web services.

File system

GEL can read or write to any delimited file including those on local disks, network disks or disk arrays.
FTP

GEL can upload or download to FTP servers.

JDBC

GEL uses JDBC to access RDBMS to read or write data.

For more information about Jelly and the Jakarta Commons Project, see http://jakarta.apache.org/commons.

GEL Setup

The GEL run-time is packaged with XOG in the XOG client. Once the client is installed, you can use the GEL command in the bin directory of the XOG client to validate and execute GEL scripts.

Make sure that JRE is installed on your computer.

GEL Script Validation and Execution

The GEL validator reads scripts, confirms that the scripts are well-formed XML, and determines that all referenced tags and tag libraries are valid and available in the runtime environment. The validator does not execute scripts.

In the following example the hello.xml script is first validated, and then the hello.xml script, located in the XOG client home directory, is executed.

E:\XOG\bin\gel -script hello.xml -validate
File geltest.xml validated.

E:\XOG\bin\gel -script hello.xml
Hello World 1!
Hello World 2!
Hello World 3!

GEL Basics

The following sections explain the basic rules for using GEL.
GEL Script Structure

The following figure shows the basic structure for a GEL script.

```
<?xml version="1.0" encoding="UTF-8"?>
    xmlns:core="jelly:core"
    xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary"
    xmlns:file="jelly:com.niku.union.gel.FileTagLibrary"
    xmlns:soap="jelly:com.niku.union.gel.SOAPTagLibrary"
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:sql="jelly:sql"
    xmlns:xog="http://www.niku.com/xog"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <!-- Code goes here -->
</gel:script>
```

Note that you can add a comment anywhere in a GEL script by using the structure `<!-- comment -->`.

GEL Script Tags

A GEL script is an executable XML file that is built from qualified elements bound to Java code called tags. Using namespace declarations, tags are organized into tag libraries which are made available in a script.

In the following Hello World example, two tag libraries are defined for the script: Core and GELTagLibrary as seen in tag pairs such as: `<core:????/>` and `<gel:????/>`.

**Note:** An entire script always resides within the GEL script tag.

Hello World Example

```
<gel:script xmlns:core="jelly:core"
    xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">
    <core:forEach indexVar='i' begin='1' end='3'>
        <gel:out>Hello World ${i}!</gel:out>
    </core:forEach>
</gel:script>
```
Variables are used extensively throughout GEL scripts. Many tags can set variables. An example of a tag that can set variables is core:set. You can use the common syntax ${variable_name} to reference variables. In the Hello World example, 'i' is a variable which is set by the forEach tag and is incremented with each loop.

Core is a built-in Jelly library that contains general-purpose tags (such as forEach that is used in the previous example). GELTagLibrary is the primary GEL library; it contains general-purpose tags not found in core and tags that are particular to CA Clarity PPM.

**Conditionals and Loops**

GEL contains the following tags for performing conditional processing:

```xml
<core:if>
  <core:if test="${hasDocs}">
    ...
  </core:if>
</core:if>

<core:choose>
  <core:choose>
    <core:when test="${row[6].equals(&quot;&quot;)}">
      ...
    </core:when>
    <core:otherwise>
      ...
    </core:otherwise>
  </core:choose>
</core:choose>

<core:switch>
  <core:switch on="${caseType}"
    <core:case fallThru="true" value="Incident"/>
    <core:case value="Problem"/>
      ...
    </core:case>
    <core:case fallThru="true" value="Question"/>
    <core:default>
      ...
    </core:default>
  </core:switch>
</core:switch>

<core:forEach>
  <core:forEach trim="true" items="${queryResult.rowsByIndex}" var="row">
    ...
  </core:forEach>
```
Variables

Variables in GEL scripts are declared at the time of use. There are no declaration blocks, like the ones you might find in other languages. GEL provides the following ways to store a variable value:

<gel:parameter>
  This tag allows values to be passed into a GEL script from a CA Clarity process. Inside the GEL script, you can refer to the parameter as you would any other variable (that is, using the ${variablename} syntax). The optional attribute secure="true" causes CA Clarity to hide the actual value in the user interface with asterisks (*).
  
  <gel:parameter var="XOGUsername" default="admin"/>
  <gel:parameter var="XOGPassword" default="password" secure="true"/>
</gel:parameter>

<core:set>
  This tag is used to set basic variables; that is, ones that do not need to be extracted from an XML document. Refer to the variable using the ${variablename} syntax.
  
  <core:set value="1" var="yes"/>
  <gel:out>${yes}</gel:out>

  You can do some basic math on the variable:

  <gel:out>${yes+2}</gel:out>
</core:set>

<gel:set>
  Use this tag when it is necessary to extract the value of the variable from an XML document. This tag differs from the <core:set> tag in that it takes a select attribute which in turn requires an XPath statement. If you are unfamiliar with XPath, think of it as a hierarchy mapping of the XML document. In the example below, the select statement points the way to the Statistics node of a XOG output file.

  <gel:set asString="false"
    var="stats"/>
</gel:set>

<gel:persist>
  This tag allows you to set variables with a scope that extends beyond the current script.
</gel:persist>
The `<gel:parse>` tag is used to create an XML document in memory. This is how you will build XOG requests. The tag can be used to generate an entire XML document, or specific nodes that can later be attached into an existing XML document.

```xml
<gel:parse var="loadContent">
  <NikuDataBus xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
               xsi:noNamespaceSchemaLocation="../xsd/nikuxog_resource.xsd">
    <Header version="12.0.0.5028" action="write" objectType="resource"
            externalSource="ORACLE-FINANCIAL"/>
    <Resources>
      <Resource resourceId="abc" isActive="true">
        <PersonalInformation lastName="doe" firstName="john"
                               emailAddress="jdoe@ca.com"/>
      </Resource>
    </Resources>
  </NikuDataBus>
</gel:parse>
```

### Built-in Parameters

Custom Action GEL scripts associated with processes have the following parameters available to them:

**Object instance ID**

If no object is associated with the process, the ID is -1. Otherwise the `$gel_objectInstanceId` parameter contains the object instance ID.

**Process ID**

`$gel_processId` is the process identifier; all instances of this process share this identifier.

**Process instance ID**

`$gel_processInstanceId` is the process instance identifier; all instances have a unique value.

All built-in parameters have a "gel_" prefix and are of data type - numeric.
Things to Watch For When Using GEL

Note the following:

- GEL is case sensitive. This statement includes variable names.
- All GEL scripts are contained in XML, therefore all XML rules apply to structure, tags, and special characters.
- In the Jelly <sql:query> tag, you cannot use the less than (<) and greater than (>) operators because they are not allowed. Use BETWEEN instead of these operators or escape the special characters using &lt; or &gt;.

Using SSL with GEL

When interacting with SOAP services in GEL, you might need to take additional steps when using the secure sockets layer (SSL) with web services. If the SSL certificate in use by the web service host has been issued by a well-known certificate authority (for example, Verisign or Thawte), no additional steps might be needed provided the appropriate certificate already exists in the cacerts keystore in the Java SDK running the GEL script.

However, you might need to take additional steps to ensure the proper trust is established between the GEL script and web service host when:

- The SSL certificate is self-signed, which means you generated the certificate using your own certificate authority.
- The expiration date on a certificate issued by a well-known certificate authority has been reached.
How to Set Up a Self-signed SSL Certificate

This process explains how to set up a self-signed SSL certificate on a web-service host.

For the setup, identify the Java SDKs that will be running GEL scripts. Here is how:

- If GEL scripts are being run outside of the application using the XOG client, the first Java SDK listed in your PATH is the one running the scripts.
- If GEL scripts are running inside a process in the application, typically the Java SDK running the BG service on the application server is the one running the scripts.

To set up a self-signed SSL certificate

1. Locate the Java SDK installation directory.
   For example, C:\jdk1.5.0_17.

2. Export the SSL certificate or any updated certificate-authority certificate you need to import to a file.
   For example, mycert.cer.

3. Change directories to the Java SDK JRE security directory.
   cd c:\jdk1.5.0_17\jre\lib\security
   This directory is where the cacerts Java keystore resides. The keystore holds certificate-authority certificates used for establishing trust. The keystore password for this keystore is always changeit.

4. Import your certificate into the cacerts keystore with the Java keytool command.
   keytool -keystore cacerts -storepass changeit -import -file c:\temp\mycert.cer -trustcacerts -alias mycert
   Assign the alias value to a value not currently in use in the keystore.
   You may be prompted on whether you want to trust this certificate. If so, answer Yes.

5. If you are setting up the self-signed certificate for the BG service for GEL scripts that run in processes, restart the BG service.
   The keystore is loaded one time when the CA Clarity services are started.

Set GEL Tag Restrictions

Use the following commands to control GEL tag restriction:

admin general restrict-gel-tags
   Sets the value of the gelTagRestriction property to on.

admin general allow-gel-tags
   Sets the value of the gelTagRestriction property to off.
The property gelTagRestriction is referenced to determine if gel tags are restricted. The property is on the system element. It is an optional element.

Use the values on or off to set GEL tag restrictions for the environment. Specifying any value other than off enables GEL tag restriction. GEL tag restrictions are off by default.

**Examples**

Properties.xml file with no GEL tag restriction:
```xml
<system online="true" multiCurrency="false" licenseTypes="old"
       singleTenantMode="true"/>
```

or
```xml
<system online="true" multiCurrency="false" licenseTypes="old"
       singleTenantMode="true" gelTagRestriction="off"/>
```

Properties.xml file with GEL tags restricted:
```xml
<system online="true" multiCurrency="false" licenseTypes="old"
       singleTenantMode="false" gelTagRestriction="on"/>
```

**Examples for Running the XOG**

By including the SOAP and XOG namespaces in GEL scripts, you give GEL the ability to communicate with the XOG web service. You must package each invocation in a proper SOAP envelope.
Example 1: Individual Calls

The following example logs into CA Clarity and runs the XOG to read the list of resources. The script performs each of these actions as individual calls to the XOG server.

```xml
    xmlns:core="jelly:core"
    xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary"
    xmlns:soap="jelly:com.niku.union.gel.SOAPTagLibrary"
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:sql="jelly:sql"
    xmlns:xog="http://www.niku.com/xog"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <gel:parameter default="http://nikuvm:80" var="XOGURL"/>
    <gel:parameter default="svong" var="XOGUsername"/>
    <gel:parameter default="svong" secure="true" var="XOGPassword"/>

    <!-- Log into XOG and get a session ID -->
    <soap:invoke endpoint="${XOGURL}/niku/xog" var="auth">
        <soap:message>
            <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                xmlns:xog="http://www.niku.com/xog">
                <soapenv:Header/>
                <soapenv:Body>
                    <xog:Login>
                        <xog:Username>${XOGUsername}</xog:Username>
                        <xog:Password>${XOGPassword}</xog:Password>
                    </xog:Login>
                </soapenv:Body>
            </soapenv:Envelope>
        </soap:message>
    </soap:invoke>

    <!-- Checking whether a sessionID is returned. If not, it means that Login was unsuccessful -->
    <gel:set asString="true" select="$auth/SOAP-ENV:Envelope/SOAP-ENV:Body/xog:SessionID/text()" var="sessionID"/>
    <core:choose>
        <core:when test="${sessionID == null}"
            >Couldn't Log in. Check the username/password.</core:when>
        <core:otherwise></core:otherwise>
    </core:choose>
```

86 XML Open Gateway Developer Guide
Examples for Running the XOG

<!-- Run XOG and attach an input file... alternatively the Body section can be the NikuDataBus section of an input file -->
<soap:invoke endpoint="${XOGURL}/niku/xog" var="runresult">

  <soap:message>
    <soapenv:Envelope
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xog="http://www.niku.com/xog">
      <soapenv:Header>
        <xog:Auth>
          <xog:SessionID>${sessionID}</xog:SessionID>
        </xog:Auth>
      </soapenv:Header>
      <soapenv:Body>
        <gel:parse var="xmlindoc" file="C:\Clarity\XOG\xml\rsm_resources_read.xml"/>
        <gel:include select="$xmlindoc"/>
      </soapenv:Body>
    </soapenv:Envelope>
  </soap:invoke>

<!-- Read the output and extract some information from it -->
<gel:set asString="true"
  select="$runresult/SOAP-ENV:Envelope/SOAP-ENV:Body/NikuDataBus/XOGOutput/Status/@state"
  var="XOGoutcome"/>
<core:switch on="${XOGoutcome}">
  <core:case value="SUCCESS">
    <gel:forEach
      var="outputnode">
      <gel:out><gel:expr select="$outputnode/PersonalInformation/@displayName"/></gel:out>
    </gel:forEach>
    <gel:set asString="false"
      var="stats"/>
    <gel:out>Success!  Total number of records: <gel:expr select="$stats/@totalNumberOfRecords"/></gel:out>
  </core:case>
  <core:case value="FAILURE">
    <gel:set asString="false"
      var="stats"/>
    <gel:out>XOG failed.  Out of <gel:expr select="$stats/@totalNumberOfRecords"/> records,
           <gel:expr select="$stats/@failureRecords"/> failed. </gel:out>
  </core:case>
  <core:default>
    <gel:out>Couldn't find XOG output summary.  Please check the output file manually. </gel:out>
  </core:default>
</core:switch>
Example 2: Single Invocation

In this example, the script logs in and makes a XOG request in a single invocation. The XOG request is also included inline, which means it is included in the script instead of being retrieved from a file.
Database Operations

GEL can connect to one or more databases, and it is not limited to CA Clarity PPM databases. Both Oracle and SQL Server are supported. See the following examples.

Most connection problems stem from either login errors or JDBC issues. The following example shows a JDBC error.

E:\Clarity\XOG\bin>gel -script gelsqlexample.xml
ERROR 2005-08-31 16:45:40.549 [main] sql.SetDataSourceTag Could not load driver class:
java.lang.ClassNotFoundException: oracle.jdbc.driver.OracleDriver
java.lang.ClassNotFoundException: oracle.jdbc.driver.OracleDriver...

If you see an error like this, find the necessary JDBC classes and copy them to the GEL classpath.

Note that the GEL engine does not search the environment path for these files. GEL only looks in the lib folder (and the CA Clarity PPM server classpath, if it has been installed on the local computer). For Oracle, ojdbc14.jar should be copied to the lib folder. For SQLServer, Microsoft's JDBC drivers must be made available to the GEL engine. Copy the files msbase.jar, mssqlserver.jar, and msutil.jar to the lib directory (after you have installed the latest JDBC driver from Microsoft, or copy the files from the CA Clarity PPM\lib directory).
The following example connects to a CA Clarity database and prints out the results of a basic query.

```xml
<gel:script xmlns:core="jelly:core"
    xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary"
    xmlns:sql="jelly:sql">

    <gel:parameter default="svong" var="ClarityUser"/>
    <gel:parameter default="svong" secure="true" var="ClarityPassword"/>

    <sql:setDataSource url="jdbc:oracle:thin:@localhost:1521:NIKU"
        driver="oracle.jdbc.driver.OracleDriver"
        user="${ClarityUser}" password="${ClarityPassword}"/>

    <sql:query var="result">
        select name, unique_name from srm_projects
    </sql:query>

    <core:forEach trim="true" items="${result.rowsByIndex}" var="row">
        <core:forEach var="field" items="${row}">
            <gel:out>${field}</gel:out>
        </core:forEach>
    </core:forEach>
</gel:script>
```

The `sql:setDataSource` statement makes the connection to the database. Note the use of parameters for the login credentials. Using `gel:parameter` allows the UserID and Password to be set from within CA Clarity PPM (furthermore, the `secure="true"` declaration masks the password in the UI) if this script is called from a CA Clarity PPM process.

`sql:query` encloses the actual query, and the two `core:forEach` loops cycle through the result. The first `core:forEach` loop runs through the rows; the embedded `core:forEach` reads the columns in each row.
The results set for this code prints out one field per line. The output would look similar to the following example.

Project ABC
P001
Consumer Confidence Project
P002
John’s Super Special Project
P003

One way around this issue is to programmatically create rows of data. The following example is for a query that returns three columns per row. By using step="3", you can process one logical row at a time. Each item is referred to by using an index offset.

```xml
<core:forEach trim="true" items="$\{queryResult.rowsByIndex\}" var="row">
  <!-- 3 fields per row, so jump by 3 to build the next row -->
  <core:forEach var="field" items="$\{queryResult.columnNames\}" indexVar="i" step="3">
    <file:line>
      <file:column value="$\{row[i]\}"/>
      <file:column value="$\{row[i+1]\}"/>
      <file:column value="$\{row[i+2]\}"/>
    </file:line>
  </core:forEach>
</core:forEach>
```

File Operations

GEL can open a file (and if it is an XML file or a comma-delimited file, parse out all the nodes and attributes), read the file, and write to it. It can also perform FTP operations on files. It cannot, however, create a directory to put files in, move files around, or delete files after it is done with them. This can be a problem when working with the Documents XOG.
**Example 1: Create a Rate Matrix XOG File**

The example code below creates a rate matrix XOG file. The file opens a tab-delimited text file as input, and creates a matrixRow node for each row of input data.

```xml
<gel:script xmlns:core="jelly:core" xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary"
xmlns:file="jelly:com.niku.union.gel.FileTagLibrary">
  <gel:parameter default="niku" var="clarityUser"/>
  <gel:parameter default="nikuadmin" secure="true" var="clarityPassword"/>
  <gel:parameter default="E:\Clarity\XOG\bin" var="infolder"/>
  <gel:parameter default="E:\clarity\xog\bin\rateMatrixLoadFile.xml" var="XOGloadfile"/>
  <gel:parameter default="${infolder}\rateMatrix.tab" var="infile"/>
  <gel:formatDate format="yyyyMMdd" stringVar="today"/>

  <!-- Open up the input file -->
  <file:readFile fileName="${infile}" delimiter="\t" var="infileParsed" embedded="false"/>

  <!-- The GEL parse statement can be given the name of an XML file, or, as shown below, an XML node structure. -->
  <!-- Use it to create the main XML shell and add in non-repetitive sections like the columns section below -->
  <gel:parse var="loadContent">
    <NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:noNamespaceSchemaLocation="..\xsd\nikuxog_matrix.xsd">
      <Header action="write" externalSource="NIKU" objectType="matrix" version="12.0.0.5028"/>
      <matrices>
        <matrix defaultCurrencyCode="USD" name="D&B COST/RATE MATRIX" type="Cost/Rate">
          <columns>
            <column name="entity"/>
            <column name="department"/>
            <column name="resourceClass"/>
            <column name="transactionClass"/>
            <column name="resourceRole"/>
            <column name="resource"/>
            <column name="inputTypeCode"/>
          </columns>
          <matrixRows/>
        </matrix>
      </matrices>
    </NikuDataBus>
  </gel:parse>
```

---

92  XML Open Gateway Developer Guide
File Operations

<!-- Build the sections. Skip the headers on the first line -->
<core:forEach items="${infileParsed.rows}" var="row" indexVar="i" begin="1" step="1">
<!-- This GEL:parse statement creates the node in memory -->
<gel:parse var="matrixRowNode">
<matrixRow actualCost="${row[11]}" currencyCode="${row[12]}" entity="${row[2]}"
department="${row[3]}"
fromDate="${row[0]}" rate="${row[9]}" transactionClass="${row[5]}" resourceClass="${row[4]}"
resourceRole="${row[6]}" inputTypeCode="${row[8]}" resource="${row[7]}"
standardCost="${row[10]}"
toDate="${row[1]}"/>
</gel:parse>
<!-- GEL:set below adds the node in memory to the main XML file we’re building -->
<gel:set value="${matrixRowNode}" select="$loadContent/NikuDataBus/matrices/matrix/matrixRows"
insert="true"/>
</core:forEach>
<!-- Now write it all to a file -->
<gel:serialize fileName="${XOGloadfile}" var="${loadContent}"/>
</gel:script>

Example 2: Output Delimited Files
This example shows how to output delimited files using GEL.
<!-- Open up the output file -->
<file:writeFile fileName="${doclistfile}" delimiter=",">

Example 3: Create a File to Write in Documents for Multiple Projects
The following example creates a file to write in documents for a number of projects. It
takes a projects XOG file as input, extracting each project ID in turn and creating a
parent node for the Documents XOG.
However, note the following:
■

The Document XOG requires the internal database ID of the project, not the
UNIQUE_NAME that appears as the project ID in the input file. You must use the
GEL JDBC connection to retrieve the corresponding DBID for each project as the
Documents XOG XML file is built.

■

The Document XOG only loads data at the folder level, which means it takes a
source folder and uploads it into a target folder. If you want to upload documents
for multiple projects, the documents for each project must be placed into an
individual temporary folder. This (creating folders, copying files into them, and so
on) is not something GEL can do currently, so you need to prepare the documents
elsewhere.

Chapter 6: GEL Scripting 93


Additionally, this example illustrates another way of building the XML file using the GEL:set tag.

```xml
<g:script xmlns:core="jelly:core"
          xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary"
          xmlns:file="jelly:com.niku.union.gel.FileTagLibrary"
          xmlns:sql="jelly:sql">
  <g:parameter default="jdbc:microsoft:sqlserver://myserver:1433;DatabaseName=pmo_dev;SelectMethod=cursor"
               var="clarityURL"/>
  <g:parameter default="niku" var="clarityUser"/>
  <g:parameter default="niku" secure="true" var="clarityPassword"/>
  <g:parameter default="D:\App\pmo\xog\xml" var="infolder"/>
  <g:parameter default="${infolder}\prj_projectswrite.xml" var="infile"/>
  <g:parameter default="D:\App\pmo\xog\xml" var="XOGlogFolder"/>
  <g:parameter default="${infolder}\DocumentsXOGLoad.xml" var="docXOGloadfile"/>
  <g:parameter default="${infolder}\docslist.gel" var="doclistfile"/>
  <g:formatDate format="yyyyMMdd" stringVar="today"/>
  <sql:setDataSource url="${clarityURL}" driver="com.microsoft.jdbc.sqlserver.SQLServerDriver"
                     user="${clarityUser}" password="${clarityPassword}" var="clarityDS"/>
  <g:parse var="projectsParsed" file="${infile}"/>
  <file:writeFile fileName="${doclistfile}" delimiter=",">
    <g:formatDate format="yyyyMMdd" stringVar="today"/>
    <sql:setDataSource url="${clarityURL}" driver="com.microsoft.jdbc.sqlserver.SQLServerDriver"
                       user="${clarityUser}" password="${clarityPassword}" var="clarityDS"/>
    <g:parse var="projectsParsed" file="${infile}"/>
</file:writeFile>
  <g:parse var="docnode">
    <NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:noNamespaceSchemaLocation="../xsd/nikuxog_document.xsd">
      <Header action="write" externalSource="OS" objectType="document" version="12.0.0.5028"/>
      <Documents>
        </Documents>
    </NikuDataBus>
  </g:parse>
  <!-- Define a variable for the Documents node -->
  <g:set select="$docsParsed/NikuDataBus/Documents" var="docnode"/>
  <!-- set up a template Parent node -->
  <g:parse var="parentNode"><Parent documentLocation="" parentObjectId="" parentObjectType="Projects"/></g:parse>
</g:script>
```
<!-- Loop thru each project ID in the input file -->
<gel:forEach select="$projectsParsed/NikuDataBus/Projects/Project" var="currentPrj">
  <gel:set asString="true" select="$currentPrj/@projectID" var="currentPrjID"/>
  <!-- reset the test flag -->
  <core:set value="false" var="hasDocs"/>

  <!-- Build the XOG file for the documents. The process is to insert a copy of the node -->
  <!-- currently in memory, then modify the attributes as necessary. After that, the -->
  <!-- node in memory is reset to the current one. Also note that the Documents XOG -->
  <!-- requires the DBID of the project ...so we have to connect to Clarity first. -->
  <core:if test="$hasDocs">
    <sql:query var="prjIDquery" dataSource="${clarityDS}"
      SELECT ID FROM niku.SRM_PROJECTS WHERE UNIQUE_NAME = ?
    <sql:param value="${row[i]}"/>
    </sql:query>

    <!-- there should only be one result value... -->
    <core:forEach trim="true" items="${prjIDquery.rowsByIndex}" var="idrow">
      <core:forEach var="idfield" items="${prjIDquery.columnNames}" indexVar="j">
        <gel:set value="${parentNode}" select="$docsParsed/NikuDataBus/Documents" insert="true"/>
        <gel:set value="${infolder}docimporttemp${currentPrjID}" select="$docnode/Parent/@documentLocation"/>
        <gel:set value="${idrow[i]}" select="$docnode/Parent/@parentObjectId"/>
        <gel:set var="parentNode" select="$docnode/Parent"/>
      </core:forEach>
    </core:forEach>
  </core:if>
</gel:forEach>

<!-- Write the XOG file for documents -->
<gel:serialize fileName="${docXOGloadfile}" var="${docsParsed}"/>

<!-- Close the output file -->
</file:writeFile>
</gel:script>
Integration Processes

A process is a way to automate repetitive steps that would otherwise be performed manually through the CA Clarity user interface. A process can act on any object type. The process includes a series of steps that result in a completed end point. A process has a start step (required), an end step (required), and one or more intermediate steps. Each step included in the process performs one or more actions that moves the process toward its completion. The following actions are available:

**Manual**
- Performed by a user in the user interface.

**System**
- Completed by a CA Clarity system action.

**Job**
- Completed by running a job either scheduled or started manually from the user interface.

**Custom action**
- Available for normal process steps that include custom GEL code. These GEL snippets use tag libraries to interact with various data sources and data destinations.

You can disconnect integration processes from any specific object. This fact allows you to:

- Schedule integration processes in CA Clarity
- Initiate integration processes in real-time in one of the following ways:
  - Manually from the GUI
  - Using a XOG web service request

From a background job, real time integrations enable external applications to send data proactively. The request starts an integration process and then passes the incoming data.

**Note:** For performance reasons, the XOG web service request does not initiate integration processes for all objects. The objects that can have integration processes initiated by the XOG web service are projects and incidents.
Basic Integration Process Checklist

Use the following checklist to set up and run integration processes:

- Create the process.
- (Optional) Create groups to represent larger segments of the process.
- Create start, intermediate, and finish steps.
- Include actions on steps. GEL can be used through Custom Actions.
- Connect the steps with splits and joins.
- Validate the process.

For more information, see the Administration Guide.

- Use the web service API to start the process, then register the web service request to invoke the process.

Note: Web service requests need not conform to any product API. They need only be valid SOAP requests.

The external application maps incoming SOAP message with a process.

The SOAP listener servlet responds to incoming web service requests.

Note: The Catalog Listener and the Ad-Hoc Query listener are built-in listeners. You can register subsequent listeners in the database using an XPath expression, a target namespace, or both to match the listener to an incoming service request.
Integration Processes

Scheduled or Manual Process

ServiceVendor

Optional Update on Completion

JobInvoked

Start Process

Step 1: Read

InvokeXOG

Step 2: Write

Step 3: Finalize

End Process

The integration process is initiated from a XOG Web service invoked by the third-party vendor. Input data is taken from the request in realtime.
Chapter 7: XOG WSDL

This section contains the following topics:

- **About the WSDL** (see page 99)
- **Object WSDL** (see page 101)
- **InvokeAction WSDL** (see page 101)
- **Query WSDL** (see page 105)
- **Examples: Microsoft Visual Studio (.NET)** (see page 110)
- **Generate Supporting API** (see page 110)

**About the WSDL**

Each XOG service includes a Web Service Description Language (WSDL) file that is downloadable from the installation. The WSDL describes the available XOG services and how to communicate with them.

**Access the WSDL**

You can access the XOG WSDL on the CA Clarity PPM application server at the following URL:

```
http://<servername:port>/niku/wsd1
```

The WSDL page is an HTML page with a list of XOG service categories.

The XOG services as listed on the WSDL page fall under the following categories:

- InvokeAction
- Query
- Object

You can display a list of all included services by clicking a category link.

For example, if you click InvokeAction, you'll see the following services:

- FlushCache
- Process
Each service includes the following links:

**Service name**

Enables an HTML page that contains the WSDL to display.

**Save As**

Enables the WSDL to download as an XML file so that you can save it to disk.

The following figure shows the links for the services available under InvokeAction.

---

**Viewers for WSDL**

There are various tools you can use to read and display the WSDL in a more readable format. There are stand-alone viewers and Enterprise Application Integration (EAI) software that map data from one system to another.

**Note:** Go to [http://www.w3.org/TR/wsdl](http://www.w3.org/TR/wsdl) for more information about web services and WSDL.
Emitter Tools for the WSDL

A client program connecting to the XOG can read the WSDL to determine what functions are available on the server. Any special data types used are embedded in the WSDL file in the form of XML Schema. The XOG WSDL has been validated against the following platforms:
- Apache AXIS 1.3
- Microsoft Visual Studio 2005 (.NET)

These emitter tools can interpret the XOG WSDL to enable communication with CA Clarity as a web service.

Because the XOG WSDL is compatible with AXIS and .NET these emitter tools generate an accurate and usable proxy API from the XOG WSDL. This facilitates interaction with the XOG services within the context of AXIS or .NET.

Object WSDL

To access the Object WSDL
1. Use the following URL to go to the Server page that lists the XOG Object services:
   http://<servername:port>/niku/wsd1/Object
2. Click an object link such as Project to access the Project WSDL. Click the All Objects link to access the WSDL for all objects from a single page.

InvokeAction WSDL

To access the InvokeAction WSDL service category
1. Use the following URL to go to the Server page that lists the XOG InvokeAction services:
   http://<servername:port>/niku/wsd1/InvokeAction
2. Click FlushCache or Process to access the corresponding WSDL.
Example: Process WSDL

The following example illustrates the WSDL for the Process service:

```xml
<definitions xmlns:soap="http://schemas.xmlsoap.org/wsd/soap/
    xmlns:tns="http://www.niku.com/xog/InvokeAction"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://schemas.xmlsoap.org/wsd/"
    name="InvokeActionProcess">
    <types>
        <xsd:schema elementFormDefault="qualified"
            targetNamespace="http://www.niku.com/xog/InvokeAction">
            <xsd:complexType name="Process">
                <xsd:sequence>
                    <xsd:element name="code" type="xsd:string" minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="request" minOccurs="0" maxOccurs="1">
                        <xsd:complexType>
                            <xsd:sequence>
                                <xsd:any/>
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:element name="Process" type="tns:Process"/>
            <xsd:element name="Auth">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element minOccurs="0" name="SessionID" type="xsd:string"/>
                        <xsd:element minOccurs="0" name="Username" type="xsd:string"/>
                        <xsd:element minOccurs="0" name="Password" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="Login">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element minOccurs="1" name="Username" type="xsd:string"/>
                        <xsd:element minOccurs="0" name="Password" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="SessionID" type="xsd:string"/>
            <xsd:element name="Logout">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element minOccurs="1" maxOccurs="1" name="SessionID" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:schema>
    </types>
</definitions>
```
<xsd:complexType>
  <xsd:element name="Process"/>
  <xsd:element name="Auth"/>
  <xsd:element name="Login"/>
  <xsd:element name="LoginResponse"/>
  <xsd:element name="Logout"/>
</xsd:complexType>

<message name="Process">
  <part element="tns:Process" name="body"/>
  <part element="tns:Auth" name="header"/>
</message>
<message name="Auth">
  <part element="tns:Auth" name="header"/>
</message>
<message name="Login">
  <part element="tns:Login" name="parameters"/>
</message>
<message name="LoginResponse">
  <part element="tns:SessionID" name="body"/>
</message>
<message name="Logout">
  <part element="tns:Logout" name="parameters"/>
</message>

<portType name="ProcessPort">
  <operation name="Process">
    <input message="tns:Process"/>
  </operation>
  <operation name="Login">
    <input message="tns:Login"/>
    <output message="tns:LoginResponse"/>
  </operation>
  <operation name="Logout">
    <input message="tns:Logout"/>
  </operation>
</portType>

<binding name="ProcessSoapBinding" type="tns:ProcessPort">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="Process">
    <soap:operation soapAction="http://www.niku.com/xog/InvokeAction/Process"
      style="document"/>
    <input>
      <soap:body parts="body" use="literal"/>
      <soap:header message="tns:Auth" part="header" use="literal"/>
    </input>
  </operation>
  <operation name="Login">
    <soap:operation soapAction="http://www.niku.com/xog/InvokeAction/Login"
      style="document"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
<soap:body use="literal"/>
</output>
</operation>
<operation name="Logout">
  <input>
    <soap:body use="literal"/>
  </input>
</operation>
</binding>
<service name="ProcessService">
  <documentation>Invoke Action Process Service</documentation>
  <port binding="tns:ProcessSoapBinding" name="ProcessService">
    <soap:address location="http://<servername:port>//xog"/>
  </port>
</service>
</definitions>
<xsd:any> and Process WSDL

The request and response types of the Object WSDL are defined as <xsd:any> as seen in the Process WSDL example. However, the schema of these request/response types needs to be consistent with the corresponding XOG Object schema (as per .xsd file).

The following example shows the SOAP representation of the Project object WSDL for the ReadProject operation. The bolded DataBus document shows the correct representation of the DataBus <xsd:any> type.

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Header>
        <Auth xmlns="http://www.niku.com/xog/Object">
            <SessionID>5000156__171120a:10a241ff830:7f711143139816999</SessionID>
        </Auth>
    </soap:Header>
    <soap:Body>
        <ReadProject xmlns="http://www.niku.com/xog/Object">
            <DataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="/xsd/xog_read.xsd" xmlns="">
                <Header version="12.0.0.5028" externalSource="NIKU" />
                <Query>
                    <Filter name="subject" criteria="EQUALS">admin</Filter>
                </Query>
            </DataBus>
        </ReadProject>
    </soap:Body>
</soap:Envelope>
```

Query WSDL

The WSDL for queries is not pre-packaged. A WSDL for a query will exist only if an NSQL query was created in Studio. Depending on the number of NSQL queries already defined in Studio, you will see the corresponding numbers of WSDLs.

To access the Query WSDL

1. Use the following URL to go to the Server page that lists the XOG Query services:
   http://<servername:port>/niku/wsdl/Query
2. Click a query type link to access the corresponding WSDL.
Example: Demand for Resource Query WSDL

The following example displays the WSDL for the Demand for Resource query service:

```xml
<definitions xmlns:tns="http://www.niku.com/xog/Query"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns=http://schemas.xmlsoap.org/wsdl/
    name="DemandforResourceQuery"
    targetNamespace="http://www.niku.com/xog/Query"/>
<types>
    <xsd:schema elementFormDefault="qualified" targetNamespace="http://www.niku.com/xog/Query">
        <xsd:complexType name="DemandforResourceFilter">
            <xsd:sequence>
                <xsd:element maxOccurs="1" minOccurs="0" name="actual_hours" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="actual_hours_from" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="actual_hours_to" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="allocated_hours" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="allocated_hours_from" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="allocated_hours_to" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="estimated_effort" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="estimated_effort_from" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="estimated_effort_to" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="resource_name" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="resource_name_from" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="resource_name_to" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="resource_name_wildcard" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="resource_id" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="resource_id_from" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="resource_id_to" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="calendar_time_key" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="calendar_time_key_from" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="calendar_time_key_to" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="calendar_time_key_wildcard" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="calendar_time_level" type="xsd:string"/>
                <xsd:element maxOccurs="1" minOccurs="0" name="calendar_time_level_from" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:schema>
</types>
```
<xsd:element maxOccurs="1" minOccurs="0" name="calendar_time_level_to" type="xsd:string"/>
<xsd:element maxOccurs="1" minOccurs="0" name="calendar_time_level_wildcard" type="xsd:string"/>
<xsd:element maxOccurs="1" minOccurs="0" name="calendar_time_label" type="xsd:string"/>
<xsd:element maxOccurs="1" minOccurs="0" name="calendar_time_label_from" type="xsd:string"/>
<xsd:element maxOccurs="1" minOccurs="0" name="calendar_time_label_to" type="xsd:string"/>
<xsd:element maxOccurs="1" minOccurs="0" name="calendar_time_label_wildcard" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="DemandforResourceQuery">
<xsd:sequence>
<xsd:element maxOccurs="1" minOccurs="1" name="Code" type="xsd:string"/>
<xsd:element maxOccurs="1" minOccurs="0" name="Filter" type="tns:DemandforResourceFilter"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="Query" type="tns:DemandforResourceQuery"/>
<xsd:complexType name="DemandforResourceRecord">
<xsd:sequence>
<xsd:element maxOccurs="1" minOccurs="1" name="actual_hours" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="1" name="allocated_hours" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="1" name="estimated_effort" type="xsd:decimal"/>
<xsd:element maxOccurs="1" minOccurs="1" name="resource_name" type="xsd:string"/>
<xsd:element maxOccurs="1" minOccurs="1" name="resource_id" type="xsd:long"/>
<xsd:element maxOccurs="1" minOccurs="1" name="calendar_time_key" type="xsd:string"/>
<xsd:element maxOccurs="1" minOccurs="1" name="calendar_time_label" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="DemandforResourceRecords">
<xsd:sequence>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="Record" type="tns:DemandforResourceRecord"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="DemandforResourceQueryResult">
<xsd:sequence>
<xsd:element maxOccurs="1" minOccurs="1" name="Code" type="xsd:string"/>
<xsd:element maxOccurs="1" minOccurs="1" name="Records" type="tns:DemandforResourceRecords"/>
</xsd:sequence>
</xsd:complexType>
<xsd:sequence>
  <xsd:complexType>
    <xsd:element name="QueryResult" type="tns:DemandforResourceQueryResult"/>
  </xsd:complexType>
</xsd:element>
</xsd:complexType>
</xsd:element>
</xsd:complexType>
</xsd:element>
</xsd:complexType>
</xsd:element>
</xsd:complexType>
</xsd:element>
</xsd:complexType>
</xsd:element>
</xsd:complexType>
</xsd:element>
</xsd:complexType>
</xsd:element>
</xsd:complexType>
</xsd:element>
</xsd:schema>

<message name="Query">
  <part element="tns:Query" name="body"/>
  <part element="tns:Auth" name="header"/>
</message>

<message name="QueryResult">
  <part element="tns:QueryResult" name="body"/>
</message>

<message name="Auth">
  <part element="tns:Auth" name="header"/>
</message>

<message name="Login">
  <part element="tns:Login" name="parameters"/>
</message>

<message name="LoginResult">
  <part element="tns:SessionID" name="body"/>
</message>

<message name="Logout">
  <part element="tns:Logout" name="parameters"/>
</message>
<portType name="DemandforResourceQueryPort">
    <operation name="Query">
        <input message="tns:Query"/>
        <output message="tns:QueryResult"/>
    </operation>
    <operation name="Login">
        <input message="tns:Login"/>
        <output message="tns:LoginResult"/>
    </operation>
    <operation name="Logout">
        <input message="tns:Logout"/>
    </operation>
</portType>

<binding name="DemandforResourceQuerySoapBinding" type="tns:DemandforResourceQueryPort">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="Query">
        <input>
            <soap:body parts="body" use="literal"/>
            <soap:header message="tns:Auth" part="header" use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
    </operation>
    <operation name="Login">
        <soap:operation soapAction="http://www.niku.com/xog/Query/Login" style="document"/>
        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
    </operation>
    <operation name="Logout">
        <input>
            <soap:body use="literal"/>
        </input>
    </operation>
</binding>

<service name="DemandforResourceQueryService">
    <documentation>Demand for Resource Query Service</documentation>
    <port binding="tns:DemandforResourceQuerySoapBinding"
        name="DemandforResourceQueryService">
        <soap:address location="http://<servername:port>://xog"/>
    </port>
</service>
Examples: Microsoft Visual Studio (.NET)

The following example is a client-side Windows Form written in .NET Visual Basic that uses the ClarityDotNetXOG API to invoke the FlushCache XOG service on a Button Click event.

```vbnet
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim xog As ClarityDotNetXOG
    xog = New ClarityDotNetXOG
    xog.Username = "admin"
    xog.Password = "admin"
    Dim ids(0 To 1) As String
    Dim groups(0 To 1) As String
    ids(0) = "ConfigurationProperties"
    groups(0) = "Resources"
    xog.FlushCache(ids, groups)
    MessageBox.Show("Flush Cache Complete")
End Sub
```

Generate Supporting API

Both Apache AXIS and Microsoft Visual Studio have emitter tools that generate proxy classes based on service descriptions that conform to the WSDL.
Generate a Proxy API from Axis

The emitter tool packaged with Apache AXIS is WSDL2Java. The following example shows how to initiate this tool from the command line against the All Objects XOG WSDL.

Important! You must include the -W flag when generating the proxy API from the XOG WSDL definitions. This indicates that the WSDL is of style: document/literal. If you do not include the -W flag, it is assumed that the WSDL is of style: wrapped/literal, which is incorrect. Omitting the flag will not throw an error in the proxy generation, but the resulting API will cause runtime errors when trying to communicate with the XOG interfaces.

The default output location of the proxy files follows the namespace convention defined in the WSDL. Because the targetNamespace defined in the All Object WSDL is http://www.niku.com/xog/Object, the resulting proxy classes from the command-line request reside in D:\axis\com\niku\www\xog\Object.

Add a Web Reference from Microsoft Visual Studio

The emitter tool packaged with Microsoft Visual Studio is implemented as a web reference. The reference is developed from within a project within the Microsoft Visual Studio GUI.

To develop a web reference from Microsoft Visual Studio

1. From Project, select Add Web Reference.
2. In URL, enter the desired XOG WSDL URL and click Go.
   The WSDL is located.
3. In Web Reference Name, enter a name for the service and click Add Reference.
   
   Note: The generated proxy API for the All Objects Web Reference is on the left side of the page. This enables a developer to communicate programmatically with the services defined in the All Objects Web Reference.
Appendix A: XOG Object Reference

Stock XOG Object Summary

You can process the following data objects or transactions using the XOG.

## Base XOG Objects

<table>
<thead>
<tr>
<th>Object or Transaction</th>
<th>In (Write)</th>
<th>Out (Read)</th>
<th>Scope</th>
<th>Master System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td>Yes</td>
<td>Yes</td>
<td>Multiple actions Step-level escalations Multiple objects Sub-processes</td>
<td>CA Clarity PPM</td>
</tr>
<tr>
<td>Content Pack</td>
<td>Yes</td>
<td>Yes</td>
<td>The transactions include: Read and write the filter portlet type with its fields. Read and write page and tab level metadata concerning filter portlet instance properties. Read and write page and tab level metadata concerning grid or graph portlet filter mappings.</td>
<td>CA Clarity PPM</td>
</tr>
<tr>
<td>customObjectInstance</td>
<td>Yes</td>
<td>Yes</td>
<td>Custom object instance</td>
<td>None</td>
</tr>
<tr>
<td>group</td>
<td>Yes</td>
<td>Yes</td>
<td>Groups</td>
<td>None</td>
</tr>
<tr>
<td>User</td>
<td>Yes</td>
<td>Yes</td>
<td>Basic User Properties OBS association fields</td>
<td>Enterprise systems</td>
</tr>
<tr>
<td>Location</td>
<td>Yes</td>
<td>Yes</td>
<td>Location properties Association with departments</td>
<td>CA Clarity PPM</td>
</tr>
<tr>
<td>Resource</td>
<td>Yes</td>
<td>Yes</td>
<td>Basic and non-labor resources properties Basic management properties Financial properties Resource contact properties</td>
<td>Enterprise systems</td>
</tr>
</tbody>
</table>
# Product Stock XOG Object Summary

<table>
<thead>
<tr>
<th>Object or Transaction</th>
<th>In (Write)</th>
<th>Out (Read)</th>
<th>Scope</th>
<th>Master System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benefit Plan</td>
<td>Yes</td>
<td>Yes</td>
<td>Financial benefit plans</td>
<td>Accounting systems</td>
</tr>
<tr>
<td>Budget Plan</td>
<td>Yes</td>
<td>Yes</td>
<td>Financial budget plans</td>
<td>Accounting systems</td>
</tr>
<tr>
<td>Capacity Planning Scenario</td>
<td>Yes</td>
<td>Yes</td>
<td>Public or private scenarios that contain an arbitrary number of project members or expression members with an arbitrary number of terms</td>
<td>Enterprise systems</td>
</tr>
<tr>
<td>Cost plan</td>
<td>Yes</td>
<td>Yes</td>
<td>Financial cost plans</td>
<td>Accounting systems</td>
</tr>
<tr>
<td>Department</td>
<td>Yes</td>
<td>Yes</td>
<td>Department properties</td>
<td>CA Clarity</td>
</tr>
<tr>
<td>Entity</td>
<td>Yes</td>
<td>Yes</td>
<td>Basic entity properties</td>
<td>CA Clarity</td>
</tr>
<tr>
<td>Financial Transaction</td>
<td>Yes</td>
<td>Yes</td>
<td>Financial Transactions</td>
<td>Accounting systems</td>
</tr>
<tr>
<td>General Ledger Account</td>
<td>Yes</td>
<td>Yes</td>
<td>GL Accounts</td>
<td>Accounting systems</td>
</tr>
<tr>
<td>General Ledger Allocation Rule</td>
<td>Yes</td>
<td>Yes</td>
<td>Standard GL allocation rules and credit rules, Investment-specific GL allocation debit rules</td>
<td>Accounting systems</td>
</tr>
<tr>
<td>General Ledger Period</td>
<td>Yes</td>
<td>No</td>
<td>GL periods</td>
<td>Enterprise</td>
</tr>
<tr>
<td>General Ledger transaction</td>
<td>No</td>
<td>Yes</td>
<td>GL transactions</td>
<td>Accounting systems</td>
</tr>
<tr>
<td>Inbound Transaction</td>
<td>Yes</td>
<td>Yes</td>
<td>Financial transactions</td>
<td>None</td>
</tr>
<tr>
<td>Investment</td>
<td>None</td>
<td>None</td>
<td>Used for import and export of investment objects (for example, asset, application, project, and so on).</td>
<td>CA Clarity</td>
</tr>
<tr>
<td>Invoices</td>
<td>Yes (status)</td>
<td>Yes</td>
<td>Invoices</td>
<td>Enterprise systems</td>
</tr>
<tr>
<td>Non-Project investment</td>
<td>Yes</td>
<td>Yes</td>
<td>Includes non-project investment objects such as assets, applications, products, and so on.</td>
<td>None</td>
</tr>
<tr>
<td>outboundTransaction</td>
<td>No</td>
<td>Yes</td>
<td>Exports financial transactions from other systems</td>
<td>None</td>
</tr>
<tr>
<td>Portfolio</td>
<td>Yes</td>
<td>No</td>
<td>import and export portfolios</td>
<td>None</td>
</tr>
<tr>
<td>Object or Transaction</td>
<td>In (Write)</td>
<td>Out (Read)</td>
<td>Scope</td>
<td>Master System</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>------------</td>
<td>-------</td>
<td>---------------</td>
</tr>
<tr>
<td>Project</td>
<td>Yes</td>
<td>Yes</td>
<td>Project schema: export project data including tasks, assignments, custom fields, management and financial properties, OBS association fields import basic properties, tasks, assignments, management properties, financial properties, custom fields, and OBS association fields Participants and participant group import</td>
<td>ERP system Timesheet system Oracle Financials Accounting system</td>
</tr>
<tr>
<td>Requisition</td>
<td>Yes</td>
<td>Yes</td>
<td>Import/export requisitions</td>
<td>None</td>
</tr>
<tr>
<td>Role</td>
<td>Yes</td>
<td>Yes</td>
<td>Basic role information Non-labor roles</td>
<td>Enterprise systems</td>
</tr>
<tr>
<td>Skill</td>
<td>Yes</td>
<td>Yes</td>
<td>Skills hierarchy</td>
<td>None</td>
</tr>
<tr>
<td>Type Code</td>
<td>Yes</td>
<td>Yes</td>
<td>Typecodes</td>
<td>None</td>
</tr>
</tbody>
</table>
Application

Use the application XOG object to view inbound and outbound application object instances.

Schema Name

nikuxog_application.xsd

Read and Write XML Files

The following XML files are included:

- inv_applications_read.xml. Use this file to export application object instances from applications_write.xml. Use this file to import application object instances that were previously exported from CA Clarity.

Prerequisites

The prerequisites (if any) are the same as that of other non-project investment objects (NPIO).

Business Rules and Processing

The business rules and processing are the same as that of other NPIOs.

Read Filters

The read filters are the same as that of other NPIOs.

Error Handling

The error handling is the same as that of other NPIOs.
Asset

Use the asset XOG object to view inbound and outbound asset object instances.

Schema Name

nikuxog_asset.xsd

Read and Write XML Files

The following XML files are included:
- inv_assets_read.xml. Use this file to export asset object instances from CA Clarity.
- inv_assets_write.xml. Use this file to import asset object instances that were previously exported from CA Clarity.

Prerequisites

The prerequisites (if any) are the same as that of other non-project investment objects (NPIO).

Business Rules and Processing

The business rules and processing are the same as that of other NPIOs.

Read Filters

The read filters are the same as that of other NPIOs.

Error Handling

The error handling is the same as that of other NPIOs.
Benefit Plan

Use the benefit plan XOG object to view inbound and outbound financial benefit plans. Benefit plans are created for existing investments.

Schema Name

nikuxog_benefitPlan.xsd

Read and Write XML Files

The following XML files are included:

- benefitPlan_read.xml. Use this file to export financial benefit plans from CA Clarity.
- benefitPlan_write.xml. Use this file to import financial benefit plans that were previously exported from CA Clarity.

Prerequisites

Before using this XOG object, make sure the following objects exist in CA Clarity:

- Investments
- Entity
- Time periods

Business Rules and Processing

The following business rules and processing apply to this XOG object:

- A Benefit Plan object is created by setting up the benefit plan default properties.
- Plan details (line items) are added to the benefit plan.
- Existing plan detail records are not deleted.

Read Filters

The following explicit read filters are used:

code
   The code for the benefit plan.

name
   The name of the benefit plan.

investmentCode
   The investment code with which the plan is associated.

Error Handling
The following errors can be thrown:

- You must have plan XOG access rights to perform this action.
- Plan code is required.
- Investment code is missing or invalid.
- Period type cannot be changed once defined.
- Investment must be associated to an entity before setting up financial plans.
- Period date is missing or invalid.
- Finish time period is invalid.
- Start time period is invalid.
- Cannot modify plan periods prior to the freeze date.
- Valid fiscal period does not exist for start and end dates for plan detail.

**Schema Mapping**

Mappings for the following schema tag names are provided:

- BenefitPlan (see page 119)
- Detail (see page 121)
- Description (see page 121)
- Segment (see page 122)

**BenefitPlan Schema Tag**

The BenefitPlan tag is part of the schema mapping for the benefit plan XOG object. It has the following attributes:

**Code**

- Required. Defines the unique code of the benefit plan.
  
  **Table and Column:** CODE
  
  **Type:** String

**FinishPeriod**

- Required. Defines the finish time period name.
  
  **Table and Column:** END_PERIOD_ID
  
  **Type:** String
**Investment Code**

Required. Defines the Investment code.

*Table and Column:* OBJECT_ID  
*Type:* String

**Investment Type**

Required. Defines the investment type.

*Table and Column:* OBJECT_CODE  
*Type:* String

**Name**

Required. Defines the benefit plan name.

*Table and Column:* NAME  
*Type:* String

**Period Type**

Defines the time period type (for example, monthly).

*Table and Column:* PERIOD_TYPE_CODE  
*Type:* String

**Revision**

Used only in XOG read result. Represents the revision of the budget plan.

*Table and Column:* REVISION  
*Type:* Integer

**StartPeriod**

Required. Defines the start time period name.

*Table and Column:* START_PERIOD_ID  
*Type:* String
**Description Schema Tag**

This schema tag is part of the schema mapping for the Benefit Plan XOG object. This schema tag is a text node.

The following attribute is included in the schema tag:

**Description**

 Defines the benefit plan description.

*Table and Column:* description

*Type:* String

**Detail Schema Tag**

The detail tag is part of the schema mapping for the benefit plan XOG object. This tag defines a benefit plan detail row and is composed primarily of schema tags.

The Detail schema tag includes the following schema tags:

**Benefit**

Defines the benefit for the specific time period segments.

*Table and Column:* BENEFIT

**ActualBenefit**

Defines the actual benefit for the specific time period segments.

*Table and Column:* ACTUAL_BENEFIT

**Custom Information**

Defines the field names for the custom information.

The Detail schema tag includes the following attribute:

**detailName**

Required. Defines the name of the benefit plan detail row.

*Table and Column:* DETAIL

*Type:* String
**Segment Schema Tag**

This tag is part of the schema mapping for the Benefit Plan XOG object. The Benefit and ActualBenefit tags can include one or more segments. This tag has the following attributes:

- **start**
  Defines the start period for the benefit or actual benefit.
  
  **Type**: String

- **finish**
  Defines the end period for the benefit or actual benefit.
  
  **Type**: String

- **value**
  Defines the value for the benefit or actual benefit.
  
  **Type**: String
Budget Plan

Use the budget plan XOG object to view inbound and outbound budget plans. A budget plan is created for an existing investment. The structure of a budget plan must meet the following criteria:

- When the forceReplace value is true, the budget plan must match the associated source cost plan.
- When the Replace value is false, the budget plan must match the associated source cost plan and the latest approved budget. If there is no latest approved budget, the structure must match the current cost plan of record.

Schema Name

nikuxog_budgetPlan.xsd

Read and Write XML Files

The following XML files are included:

- budgetPlan_read.xml. Use this file to export budget plans from CA Clarity.
- budgetPlan_write.xml. Use this file to import budget plans that were previously exported from CA Clarity.

Prerequisites

Before using this XOG object, make sure the following objects exist in CA Clarity:

- Investments
- Entity
- Time periods
- Details used in the plan
**Business Rules and Processing**

The following business rules and processing apply to this XOG object:

- A budget plan uses the Cost Plan object internally.
- The budget plan structure must match the cost plan of record structure and potentially the latest approved budget plan structure, depending on the value of the forceReplace attribute.
- Cost plan details (line items) are added to the budget plan.
- Existing plan detail records are not deleted.

**Read Filters**

The following explicit read filters are used:

- code. The code of the budget plan.
- name. The name of the budget plan.
- investmentCode. The investment code with which the plan is associated.

**Error Handling**

The following error messages can be thrown:

- You must have Budget Plan - XOG Access Rights to perform this action.
- Plan code is required.
- Grouping attribute is missing or invalid.
- Investment code is missing or invalid.
- Period type cannot be changed once defined.
- Grouping attributes cannot be changed once defined.
- Investment must be associated to an entity before setting up financial plans.
- Benefit plan ID is missing or invalid.
- Period date is missing or invalid.
- Grouping attributes in plan details do not match plan grouping attributes.
- GL account is missing or invalid.
- Missing or invalid value for grouping attribute.
- Finish time period is invalid.
- Start time period is invalid.
- Grouping attributes do not match locked plan structure for associated entity.
- Cannot modify plan periods prior to the freeze date.
- Valid fiscal period does not exist for start and end dates for plan detail.
- Submit for Approval requires a cost plan of record.
- A submitted budget already exists.
- You cannot submit a plan for approval whose total cost is zero.
- The structure of the existing budget and the cost plan of record do not match.

**Schema Mapping**

Mappings for the following schema tag names are provided:

- **BudgetPlan** (see page 125)
- **Description** (see page 127)
- **Grouping Attributes** (see page 128)
- **Detail** (see page 129)
- **Segment** (see page 129)

**BudgetPlan Schema Tag**

The `BudgetPlan` tag is part of the schema mapping for the Budget Plan XOG object. This tag has the following attributes:

**benefitPlanCode**

Defines the ID of the benefit plan that is associated with the budget plan.

*Table and Column:* BENEFIT_PLAN_ID

*Type:* String

**code**

Required. Defines the unique code of the cost plan.

*Table and Column:* CODE

*Type:* String

**finishPeriod**

Required. Defines the finish time period name.

*Table and Column:* END_PERIOD_ID

*Type:* String
forceReplace
  Required. Specifies whether the existing budget structure and period types can be
different from the latest approved budget plan. XOG out always uses the value of
True.
  **Field:** forceReplace
  **Type:** Boolean
  **Possible Values:** True, False

investmentCode
  Required. Defines the Investment code
  **Table and Column:** OBJECT_ID
  **Type:** String

investmentType
  Defines the investment type. This attribute is used only in the XOG read result.
  **Type:** String

name
  Defines the budget plan name
  **Table and Column:** NAME
  **Type:** String

revision
  Used only in XOG read results. Represents the revision of the budget plan.
  **Field Name:** REVISION
  **Type:** Integer

periodType
  Defines the time period type.
  **Field Name:** PERIOD_TYPE_CODE
  **Type:** String

sourceCostPlanCode
  Defines the ID of the source cost plan for the submitted budget.
  **Table and Column:** SOURCE_COST_PLAN_ID
  **Type:** String
**startPeriod**

Required. Defines the start time period name.

**Field Name:** START_PERIOD_ID

**Type:** String

**status**

Defines the status of the plan.

**Possible Values:** Submitted, Approved, Rejected.

**Table and Column:** STATUS_CODE

**Type:** String

**Description Schema Tag**

This schema tag is part of the schema mapping for the Budget Plan XOG object. This is a text node. The schema tag includes the following attribute.

**Description**

Defines the Budget Plan description.

**Table and Column:** description

**Type:** String
Grouping Attributes Schema Tag

This tag is part of the schema mapping for the Budget Plan XOG object. This tag is used to define grouping attributes for a budget plan. The grouping attributes for a budget plan must match the grouping attributes of the cost plan of record for the investment.

Grouping Attribute

Defines a grouping of attributes for a budget plan.

Possible Values:

- charge_code_id
- role_id
- resource_id
- department_id
- location_id
- transaction_class_id
- resource_class_id
- input_type_code_id
- lov1_id
- lov2_id

Table and Column: The value is stored in ODF_MULTI_VALUE Attributes table.

Type: String
**Detail Schema Tag**

This tag is part of the schema mapping for the Budget Plan XOG object. This tag defines a budget plan detail row and is composed primarily of schema tags.

**Note:** The Detail schema tag also has two attributes: glAccountMain and glAccountSub. These attributes define the GL Account associated with the Detail schema tag. The attributes map to the GL_ACCOUNT_ID column.

The Detail schema tag includes the following tags:

- **Cost**
  - Defines the cost for specific time period segments.
  - **Table and Column:** COST

- **Units**
  - Defines the quantity for a specific time period segment.
  - **Table and Column:** UNITS

- **Revenue**
  - Defines the revenue for a specific time period segment.
  - **Table and Column:** REVENUE

- **Grouping Attributes**
  - Defines the grouping attribute codes and values for the grouping attributes selected for the cost plan. Each grouping attribute represents a code and value pair for an attribute.
  - **Table and Column:** The value is stored in the column corresponding to the groupingAttribute code (for example, location_id, charge_code_id).
  - **Type:** String

- **Custom Information**
  - Defines the field names for the custom information.

**Segment Schema Tag**

This tag is part of the schema mapping for the Budget Plan XOG object. The Unit, Cost, and Revenue schema tags will include one or more Segment schema tags. This tag has the following attributes:

- **start**
  - Defines the start period for the unit, cost, or revenue.
  - **Type:** String
finish

Defines the end period for the unit, cost, or revenue.

Type: String

value

Defines the value for the unit, cost, or revenue.

Type: String
Capacity Planning Scenario

Use the Capacity Planning Scenario XOG object to view inbound and outbound capacity planning scenario attributes. Scenarios are defined for inbound (write) and outbound (read) processing.

Schema Name

nikuxog_capplanScenario.xsd

Read and Write XML Files

The following XML files are included:

- caplan_scenarios_read.xml. Use this file to export capacity planning scenarios from CA Clarity.
- caplan_scenarios_write.xml. Use this file to import capacity planning scenarios that were previously exported from CA Clarity.

Prerequisites

The following conditions must be met before importing capacity planning scenarios:

- The scenario must be created by you.
- Some of the fields that a scenario can include are objects that must already exist in CA Clarity in order to be imported. These include projects identified by SRM_PROJECTS.UNIQUE_NAME.
- If CA Clarity is built using XOG imports, charge codes and projects must be imported before importing capacity planning scenarios.
- The maximum number of OBS levels you can XOG is 10.

Read Filters

The XOG processes outbound capacity planning schemas based on the following field:

ownerID

The ID of a valid user (CMN_SEC_USERS.ID).

Error Handling

The following fields are written to the Success and Error files when the XOG process generates an error or warning:

- ID. For all errors, the scenario ID is posted to the Success and Error files.
- Name. For all errors, the scenario name is posted to the Success and Error files.

Schema Mappings
Mappings for the following schema tag names are provided.

- Scenario
- Segment

**Scenario Schema Tag**

This tag is part of the schema mapping for the capacity planning scenario XOG object and is composed of the Member element.

The Scenario schema tag has the following attributes:

**name**
- Required. Defines the scenario name. This does not have to be unique.
- **Table and Column**: CAP_SCENARIOS.NAME
- **Type**: String

**ownerUserName**
- Required. Defines the user who owns the scenario. The user_id is found in the CMN_SEC_USERS table.
- **Table and Column**: CAP_SCENARIOS.USER_ID
- **Type**: Number

**description**
- Describes the scenario.
- **Table and Column**: CAP_SCENARIOS.DESCRIPTION
- **Type**: String

**budgetBenefit**
- Contains the scenario (that is, what-if) value for benefit amount.
- **Table and Column**: CAP_SCENARIOS.BDGT_CST_TOTAL
- **Type**: Number

**budgetCost**
- Contains the scenario (that is, what-if) amount for budget cost.
- **Table and Column**: CAP_SCENARIOS.BDGT_REV_TOTAL
- **Type**: Number
isPublic

Currently not used.

Table and Column: CAP_SCENARIOS.IS_PUBLIC
Type: Number

Member Element

The Member is the sub-element of the Scenario schema tag. It is composed of the following elements:

- Expression. A power filter expression that determines which investments to include.
- Investment. A specific instance of an investment to include in the scenario.

The member element has the following attributes:

investmentType

Required. Defines the type of investment.

Table and Column: CAP_SCENARIO_MEMBERS.MEMBER_TYPE
Type: String

isExcluded

Required. Indicates if the member is excluded or not considered in this scenario.

Values:

- True
- False

Table and Column: CAP_SCENARIO_MEMBERS.IS EXCLUDED
Type: Boolean

isActive

Required. Indicates if the member is included in the scenario or hidden.

Values:

- True
- False

Table and Column: CAP_SCENARIO_MEMBERS.IS ACTIVE
Type: Boolean
isApproved
Required. Indicates if the member is considered approved in this scenario.
Values:
- 1. True
- 0. False
Table and Column: CAP_SCENARIO_MEMBERS.IS_APPROVED
Type: Boolean

priority
Required. Defines the priority.
Values: 0-36, where:
- 0. Highest priority
- 36. Lowest priority
Table and Column: CAP_SCENARIO_MEMBERS.PRIORITY
Type: Integer

lastSyncDate
Required. Defines the last time an expression was synchronized against the pool of available investments.
Table and Column: CAP_SCENARIO_MEMBERS.LAST_SYNC_DATE
Type: Date

**Expression Element**

This is the sub-element of the Member element. The Expression element has the following attribute:

**Expression**
Required. Defines the text view of the ODF-based power filter.
Table and Column: ODF_FILTER_EXPRESSIONS.EXPRESSION
Type: String

**Investment Element**

This is the sub-element of the Member element. The Investment element is composed of the following elements:
- Resources. A list of resources allocated to the investment.
- Tasks. A list of tasks and assignments for the investment.

The Investment element has the following attributes:
start
Defines the start date for the investment.
Table and Column: CAP_SCENARIO_MEMBERS.START_DATE
Type: Date

finish
The finish date for the investment.
Table and Column: CAP_SCENARIO_MEMBERS.FINISH_DATE
Type: Date

investmentID
Required. Defines the criteria for specifying the investment.
The OBJECT_ID contains the actual key for the investment.
Table and Column: CAP_SCENARIO_MEMBERS.OBJECT_ID
Type: Number

Resource Element

This is a sub-element of the Investment element. Resource is composed of the AllocCurve (Segment) element which is a time-scaled value.

The Resource element has the following attributes:

bookingStatus
Required. The booking status of the resource (hard, soft, or mixed).
Table and Column: CAP_SCENARIO_TEAM.PRBOOKING
Type: Number

defaultAllocation
Required. Defines the repeating segments that represent a resource's allocation to an investment (that is, start, finish, and allocation percentage) stored in the allocation curve.
Table and Column: CAP_SCENARIO_TEAM.PRALLOCCURVE
Type: Number

projectRoleID
Required. Defines the role ID or key.
Table and Column: CAP_SCENARIO_TEAM.PRROLEID
Type: Number
resourceID

Required. Defines the resource ID or key.

**Table and Column:** CAP_SCENARIO_TEAM.PRRESOURCEID

**Type:** Number

### Segment Schema Tag

This tag is part of the schema mapping for the capacity planning scenario XOG object. Segment is also known as the Time Scale Value.

The Segment schema tag has the following attributes:

**finish**

Required. Stored as BLOB.

**Table and Column:** PRALLOCCURVE

**Type:** Date

**start**

Required. Stored as BLOB.

**Table and Column:** PRALLOCCURVE

**Type:** Date

**sum**

Required. Stored as BLOB.

**Table and Column:** PRALLOCCURVE

**Type:** Float
Task Element

Task is composed of the Assignments (TaskLabor) element, a list of the resources assigned to the task.

The Task element has the following attributes:

- **duration**
  Defines the duration of the task in the scenario.
  
  **Table and Column:** CAP_SCENARIO_TASKS.DURATION  
  **Type:** Number

- **start**
  Required. Defines the start date of the task in the scenario.
  
  **Table and Column:** CAP_SCENARIO_TASKS.START_DATE  
  **Type:** Date

- **finish**
  Required. Defines the finish date of the task in the scenario.
  
  **Table and Column:** CAP_SCENARIO_TASKS.FINISH_DATE  
  **Type:** Date

- **name**
  Output only. Derived from PRTASK.PRNAME.
  
  **Table and Column:** N/A  
  **Type:** String

- **milestone**
  Required. Indicates if the task is a milestone.
  
  **Values:**
  - 1. True
  - 0. False
  
  **Table and Column:** CAP_SCENARIO_TASKS.IS_MILESTONE  
  **Type:** Boolean

- **taskID**
  Defines the link to a specific task on the investment.
  
  **Table and Column:** CAP_SCENARIO_TASKS.EXTERNAL_ID  
  **Type:** String
EstCurve (Segment) Element

Segment (EstCurve) element is also known as Time Scale Value. The EstCurve element has the following attributes:

**finish**
- Required. This is stored as BLOB.
- **Table and Column:** CAP_SCENARIO_ASSIGNMENTS.PREXTENSION
- **Type:** Date

**start**
- Required. This is stored as BLOB.
- **Table and Column:** CAP_SCENARIO_ASSIGNMENTS.PREXTENSION
- **Type:** Date

**sum**
- Required. Defines the amount of work remaining. It is stored as BLOB.
- **Table and Column:** CAP_SCENARIO_ASSIGNMENTS.PREXTENSION
- **Type:** Float

TaskLabor Element

TaskLabor is composed of the Segment (EstCurve) element, which is a list of segments inside the Estimate Curve. The TaskLabor has the following attributes:

**start**
- Required. Defines the start date of the task in the scenario.
- **Table and Column:** CAP_SCENARIO_ASSIGNMENTS.START_DATE
- **Type:** Date

**remainingWork**
- Derived from the assignment curve.
- **Table and Column:** CAP_SCENARIO_ASSIGNMENTS.PREXTENSION
- **Type:** Number

**actualWork**
- Derived from the assignment curve.
- **Table and Column:** CAP_SCENARIO_ASSIGNMENTS.PREXTENSION
- **Type:** Number
resourceID

Required. Defines the resource ID or key assigned to the task.

*Table and Column:* CAP_SCENARIO_ASSIGNMENTS.RESOURCE_ID

*Type:* Number
Change Request

Use the change request XOG object to view inbound and outbound change request instances.

Schema Name

nikuxog_change.xsd

Read and Write XML Files

The following XML files are included:
- change_read.xml. Use this file to export change request instance from CA Clarity.
- change_write.xml. Use this file to import change request instances that were previously exported from CA Clarity.

Prerequisites

Before using this XOG object, make sure the referenced objects, such as the project, user, and category, exist in CA Clarity.

Read Filters

The following explicit read filters are used:

- **projectCode**
  - Defines the code for the associated project.

- **Name**
  - Defines the name of the change request.

- **riskCode**
  - Defines the risk of the change request.

- **statusCode**
  - Defines the status of the change request.

- **priorityCode**
  - Defines the priority of the change request.

- **ownerCode**
  - Defines the name of the owner or assignee of the change request.

Error Handling

The following errors can be thrown:
- Assessor does not exist in the system.
- Approved By does not exist in the system.
- Project does not exist in the system.
- Category type is not valid.
- Status is not valid.
- Priority is not valid.
- Approach code is not valid.
- Owner does not exist in the system.
- Impact is not valid.
- Probability is not valid.
- Resolved By does not exist in the system.
- Task does not exist for the given project.
- Failed to import risk/issue/change request.

**Schema Mapping**

Mappings for the following schema tag names are provided:
- Change Request

**Change Request Schema Tag**

The change request tag is part of the schema mapping for the change request XOG object. It has the following attributes:

- **name**
  - Required. Defines the name of the change request.
  - **Table and Column:** NAME
  - **Type:** String

- **code**
  - Required. Defines the unique identifier for this change request.
  - **Table and Column:** RIM_RISK_ISSUE_CODE
  - **Type:** String

- **projectCode**
  - Required. Defines the project associated with this change request.
  - **Table and Column:** INV_INVESTMENTS.CODE
  - **Type:** String
approvedBy

Defines the name of the resource who has approved the request.

**Table and Column:** APPROVED_BY

**Type:** String

approvedDate

Defines the date the request was approved.

**Table and Column:** APPROVED_DATE

**Type:** Date

assessmentDate

Defines the date the request was assessed.

**Table and Column:** ASSESSMENT_DATE

**Type:** Date

assessor

Defines the name of the resource who assessed the request.

**Table and Column:** ASSESSOR

**Type:** String

ownerCode

Required. Defines the name of the resource assigned to this change request.

**Table and Column:** ASSIGNED_TO

**Type:** String

benefits

Defines the benefits of this request.

**Table and Column:** BENEFITS

**Type:** String

categoryTypeCode

Defines the category of this request.

**Table and Column:** CATEGORY_TYPE_CODE

**Type:** String

effectOnCost

Change in cost (if any) for this request

**Table and Column:** EFFECT_ON_COST

**Type:** Money
**effectOnResources**
Change in resources (if any) for this request

**Table and Column:** EFFECT_ON_RESOURCES
**Type:** Integer

**effectOnSchedule**
Defines the effect on the schedule in days for this request.

**Table and Column:** EFFECT_ON_SCHEDULE
**Type:** number (floating point)

**closureDate**
Defines the date this request was closed.

**Table and Column:** CLOSURE_DATE
**Type:** String

**description**
Defines the description of this change request.

**Table and Column:** DESCRIPTION
**Type:** String

**targetResolutionDate**
Defines the date this change request is targeted to close.

**Table and Column:** TARGET_RESOLUTION_DATE
**Type:** Date

**impactBaseline**
Defines the impact this change request has on the baseline.

**Table and Column:** IMPACT_ON_BASELINE
**Type:** String

**impactDescription**
Defines the impact this change request has on other projects.

**Table and Column:** IMPACT_DESCRIPTION
**Type:** String

**priorityCode**
Defines the priority of this change request.

**Table and Column:** PRIORITY_CODE
**Type:** String
**reasons**

Defines the reason for the change request.

*Table and Column:* ASSUMPTIONS  
*Type:* String

**reviewDate**

Defines the date the request was reviewed.

*Table and Column:* REVIEW_DATE  
*Type:* Date

**statusCode**

Defines the status of this change request.

*Table and Column:* STATUS_CODE  
*Type:* String
Charge Code

Use the charge code XOG object to view inbound and outbound charge code instances.

Schema Name

nikuxog_chargecode.xsd

Read and Write XML Files

The following XML files are included:
- prj_chargecodes_read.xml. Use this file to export charge codes from CA Clarity.
- prj_chargecodes_write.xml. Use this file to import charge codes that were previously exported from CA Clarity.

Prerequisites

None.

Read Filters

The following explicit read filters are used:
- `open`
  Specifies whether the charge code has a status of "Open".
- `chargeCodeID`
  Defines the unique identifier for the charge code.

Error Handling

The following errors can be thrown:
- Project does not exist in the system.
- Cannot change the project for the charge code.
- Failed to import Charge Codes.

Schema Mapping

Mappings for the following schema tag name is provided:
- Charge Code
Charge Code (Chargecode) Schema Tag

The charge code tag is part schema mapping for the charge code XOG object. It has the following attributes:

chargeCodeID
   Required. Defines the charge code's unique identifier.
   
   **Table and Column:** CHARGECODE
   **Type:** String

name
   Required. Defines the charge code's name
   
   **Table and Column:** NAME
   **Type:** String

openForTimeEntry
   Not required. Defines the charge code open status.
   
   **Table and Column:** OPENFORTIMEENTRY
   **Type:** Boolean

allocationStatus
   Not required. Defines the allocation status.
   
   **Values:** ACTIVE, INACTIVE, ONHOLD
   **Default:** INACTIVE
   
   **Table and Column:** ALLOCATIONSTATUS
   **Type:** String

projectCode
   Not required. Defines the code of the project associated with this charge code.
   
   **Table and Column:** PROJECTCODE
   **Type:** String
Company

Use the company XOG object to view inbound and outbound company attributes. Companies are defined for inbound (write) and outbound (read) processing.

Schema Name

nikuxog_company.xsd

Read and Write XML Files

The following XML files are included:

- **biz_companies_read.xml.** Use this file to export companies from CA Clarity.
- **biz_companies_write.xml.** Use this file to import companies that were previously exported from CA Clarity.

Terms

The following terms are used with Company XOG object:

**Parent and Affiliate Company**

These browse fields are used to associate a company with a parent or affiliate company and are used to perform validation against SRM_COMPANIES table.

If the company does not exist, no information is posted to the company supplemental fields and a warning is written to the Success and Error file. If the company exists, the field is populated.

**Account Manager**

This browse field is used to associate a company with a project manager and to perform validation with uniqueName in the SRM_RESOURCES table.

If the resource does not exist, no information is posted to the company supplemental fields and a warning is posted to the Success and Error file. If the resource uniqueName is contained in CA Clarity, the field is populated.

**Internal Contact**

This browse field is used to associate an internal contact with a company and perform validation on Last_Name in SRM_RESOURCES.

If the resource does not exist, a blank field is posted to company custom-defined fields and a warning is written to the Success and Error file. If the resource last_name exists, the field is populated.

**Billing Address**

The billing address of the company. A company can have more than one billing address. If more than one billing address exists, each is associated with the same company.
Custom Fields

Use these to import custom-defined fields. First generate the custom-defined fields with XDM. The XOG allows for an unlimited number of custom-defined fields if you map the generated field to the XML schema. Within the schema for custom-defined fields, you must provide the Column Name, Attribute Name, and Value.

Financial Properties

Prior to importing companies, the following financial properties must be set up in the Financial Administration module. The Default Values must be populated in the Administration Tool’s Application Administration/Financial Management/Defaults section. The Location, Department, WIP Class, Project Class, and Company Class values are not required in the XML schema but are required within CA Clarity.

Lookup values

The XML schema requires lookup codes provided later in this guide. These are validated against the values in CMN_LOOKUPS.

OBS association

There is an OBS Associations portlet that contains the OBS unit associated to the company, if any. The OBS association fields can be used for import and export.

Read Filters

The XOG supports outbound processing of companies based on the following fields:

- Company Status
- Company Type

And and Or processing is supported between these two fields (listed above) and for processing within Company Type. The following combinations are supported:

Company Status = x
    where x = Active or Inactive

Company Type = x
    where x = Prospect, Other, Competitor, Customer, Department, Marketplace Buyer, Marketplace Supplier, Resource Partner, Project Partner, Trust Client, Vendor

Company Status = x AND Company Type = y
    where x = Active or Inactive
    where y = one of many Company Types
Error Handling

The following fields are written to the Success and Error file when the XOG process generates an error or warning:

- companyId
- companyName
- externalId
- externalSource

The Company XOG object handles errors and warnings. If an error occurs, the table is not updated. You must fix the error and run the XOG again. If a warning occurs, the record is posted but the non-required fields are defaulted because of inconsistencies in the data.

The following errors are validated against Company:

**companyId**

The unique identifier for the company.

The company ID is validated against the companyId field. If the company ID is not unique, the company is not imported and an error is posted to the Success and Error file.

**parentCompany**

The name of the parent company associated to the company.

The parent company name is validated against the parentCompany field. If the parent company does not exist, the company is imported without any association to a parent company.

**affiliateCompany**

The name of the affiliate company associated to the company.

The affiliate company is validated against the affiliateCompany field. If the affiliate company does not exist, the company is imported without any association to an affiliate company.
**Schema Mappings**

The following schema mappings are provided for the company XOG:

- **Company** (see page 150)
- **Contact Information** (see page 151)
- **Supplemental Information** (see page 153)
- **Custom Information** (see page 155)
- **Financial Information** (see page 156)
- **Billing Address** (see page 158)
- **Billing Address Detail** (see page 158)
- **OBS Associations** (see page 160)

**Company Schema Tag**

The Company tag is part of the schema mapping for the Company XOG object. It has the following attributes:

- **companyId**
  - Required. Defines the unique, primary key for the company.
  - **Table and Column:** SRM_COMPANIES.Company_ID
  - **Type:** String

- **name**
  - Required. Defines the company name.
  - **Table and Column:** SRM_COMPANIES.Company_Name
  - **Type:** String

- **type**
  - Required. Defines the company type.
  - **Values:** Prospect, Other, Competitor, Customer, Department, Marketplace Buyer, Marketplace Supplier, Resource Partner, Project Partner, Trusted Client, Vendor
  - **Table and Column:** SRM_COMPANIES.Type
  - **Type:** String
status

Required. Defines the status for the company.

Values: Active and Inactive

Default: Active

Table and Column: SRM_COMPANIES.Status

Type: String

externalSource

Required by the schema. The lookup value specifies the originating system ID (for example, Oracle).

Table and Column: SRM_COMPANIES.External_Source_ID

Type: String (in schema) and Number (in CA Clarity)

devices

Contact Information Schema Tag

The Contact Information tag is part of the schema mapping for the Company XOG object. Contact information includes phone and fax numbers and mail and email addresses.

This schema tag has the following attributes:

address1

Defines the first line of the address.

Table and Column: SRM_CONTACTS.Address1

Type: String

address2

Defines the second line of the address.

Table and Column: SRM_CONTACTS.Address2

Type: String
city
Defines the city.
**Table and Column:** SRM_CONTACTS.City
**Type:** String

county
Defines the county.
**Table and Column:** SRM_CONTACTS.County
**Type:** String

state
Defines the state.
**Table and Column:** SRM_CONTACTS.State_Province
**Type:** String

postalCode
Defines the postal code.
**Table and Column:** SRM_CONTACTS.Postal_Code
**Type:** String

country
Defines the country.
**Table and Column:** SRM_CONTACTS.Country_ID
**Type:** Number

workPhone
Defines the work phone number.
**Table and Column:** SRM_CONTACTS.Phone_Work
**Type:** String

fax
Defines the fax number.
**Table and Column:** SRM_CONTACTS.Phone_Fax
**Type:** String

webAddress
Defines the Web address.
**Table and Column:** SRM_CONTACTS.URL
**Type:** String
Supplemental Information Schema Tag

This tag is part of the schema mapping for the Company XOG object. It has the following attributes:

**description**

Describes the company.

*Table and Column:* BIZ_COM_SUP_PROPERTIES.Description

*Type:* String

**rating**

Lookup values include:

- High
- Medium
- Low

*Table and Column:* BIZ_COM_SUP_PROPERTIES.Rating

*Type:* String

**sicCode**

Lookup values include:

- SIC Code 1
- SIC Code 2
- SIC Code 3

*Table and Column:* BIZ_COM_SUP_PROPERTIES.SIC_Code

*Type:* String

**parentCompany**

Browse this field to specify the parent with which the company is associated.

*Table and Column:* BIZ_COM_SUP_PROPERTIES.Parent_Company

*Type:* String

**affiliate Company**

Browse this field to specify the affiliate with which the company is associated.

*Table and Column:* BIZ_COM_SUP_PROPERTIES.Affiliate_Company

*Type:* String
division
Defines the company division.
**Table and Column:** BIZ_COM_SUP_PROPERTIES.Division
**Type:** String

category
Defines the company category.
**Table and Column:** BIZ_COM_SUP_PROPERTIES.Category
**Type:** String

industry
Defines the industry in which the company operates.
**Table and Column:** BIZ_COM_SUP_PROPERTIES.Industry
**Type:** String

numberOfEmployees
Defines the number of employees in the company.
**Table and Column:** BIZ_COM_SUP_PROPERTIES.Number_of_Employees
**Type:** Number

ownership
The Lookup value is "Corporation".
**Table and Column:** BIZ_COM_SUP_PROPERTIES.OwnerShip
**Type:** String

tickerSymbol
Defines the ticker symbol of the company
**Table and Column:** BIZ_COM_SUP_PROPERTIES.Ticker_Symbol
**Type:** String

referralSource
Defines the referral source.
**Table and Column:** BIZ_COM_SUP_PROPERTIES.Referral_Source
**Type:** String

accountManager
Browse this field to identify the account manager associated with the company.
**Table and Column:** BIZ_COM_SUP_PROPERTIES.Account_Manager
**Type:** String
primaryContactName

Defines the primary contact in the company.

**Table and Column:** BIZ_COM_SUP_PROPERTIES.Primary_Conact_Name

**Type:** String

primaryContact Email

Defines the email address of the company's primary contact.

**Table and Column:** BIZ_COM_SUP_PROPERTIES.Primary_Conact_Email

**Type:** String

primaryContact Phone

Defines the phone number of the company's primary contact.

**Table and Column:** BIZ_COM_SUP_PROPERTIES.Primary_Conact_Phone

**Type:** String

notes

Defines any company notes.

**Table and Column:** BIZ_COM_SUP_PROPERTIES.Notes

**Type:** String

---

**Custom Information Schema Tag**

The Custom Information tag is part of the schema mapping for the Company XOG object. This tag stores custom-defined fields (CDF). You must allow several CDFs for each company.

This schema tag has the following attributes:

ceoName

Defines the name of the company's Chief Executive Officer.

**Table and Column:** XDM_CDF_SRM_COMPANIES.XDM_CEO_NAME

**Type:** String

defaultWebSite

Defines the company's default Web address.

**Table and Column:** XDM_CDF_SRM_COMPANIES.XDM_DEFAULTWEBSITE

**Type:** String
**numberOfEmployees**

Defines the number of employees in the company.

*Table and Column:* XDM_CDF_SRM_COMPANIES.XDM_NUM_OF_EMPLOYEES

*Type:* Number

**opportunity**

Defines the opportunity for the company.

*Table and Column:* XDM_CDF_SRM_COMPANIES.XDM_OPPORTUNITY

*Type:* Boolean

*Default:* False

**internalContact**

Defines the name of the internal contact for the company. This is a browse field.

*Table and Column:* XDM_CDF_SRM_COMPANIES.XDM_PRIM_INTERNAL_CONTACT

*Type:* String

**agreementStartDate**

Defines the agreement start date for the company.

*Table and Column:* XDM_CDF_SRM_COMPANIES.XDM_AGREEMENT_START_DATE

*Type:* Date

**industry**

Defines the industry type for the company. This is a lookup value.

*Table and Column:* XDM_CDF_SRM_COMPANIES.XDM_INDUSTRY

*Type:* String

---

**Financial Information Schema Tag**

The Financial Information tag is part of the schema mapping for the Company XOG object. The attribute values are unlike other lookup values. They require you to provide a text string instead of a lookup code.

This tag has the following attributes:
status
 Defines the company’s status.

Values:
■ Active
■ Inactive
■ No new business

Default: Active

Table and Column: CLNTSUPP.STATUS_TYPE
Type: String

location
 Indicates the company location. This is a browse field.

Table and Column: CLNTSUPP.LOCATIONID
Type: String

department
 Indicates the department associated with the company. This is a browse field.

Table and Column: CLNTSUPP.DEPARTCODE
Type: String

wipClass
 Defines the WIP Class associated with the company. This is a browse field.

Table and Column: CLNTSUPP.CLNTWIPCLASS
Type: String

projectClass
 Defines the project class associated with the company. This is a browse field.

Table and Column: CLNTSUPP.PROJCLASS
Type: String

companyClass
 Defines the company class associated with the company. This is a browse field.

Table and Column: CLNTSUPP.COMPCLASS
Type: String
**batchCycle**

Optional. Defines the batch cycle associated with the company.

*Table and Column:* CLNTSUPP.BILLCYCLE  
*Type:* String

**dateOpened**

Defines the opened date for the company.

*Table and Column:* CLNTSUPP.OPENEDDATE  
*Type:* Date

**Billing Address Schema Tag**

The Billing Address tag is part of the schema mapping for the company XOG object. This is the Billing Address header. Each company can have one or many billing addresses.

This schema tag has the following attributes:

**billingCompanyName**

Required. Defines the Bill To company name.

*Table and Column:* CLNTBILLTO.COMPANY_CODE  
*Type:* String

**billingCode**

Required. Defines the billing code for the company.

*Table and Column:* CLNTBILLTO.BILL_TO_COMPANY_CODE  
*Type:* String

**Billing Address Detail Schema Tag**

The Billing Address Detail tag is part of the schema mapping for the company XOG object. It has the following attributes:

**address1**

Defines the first line of the billing address for the company.

*Table and Column:* ARMASTER.ADDR1  
*Type:* String
address2
Defines the second line of the billing address for the company.

Table and Column: ARMASTER.ADDR2
Type: String

address3
Defines the third line of the billing address for the company.

Table and Column: ARMASTER.ADDR3
Type: String

address4
Defines the fourth line of the billing address for the company.

Table and Column: ARMASTER.ADDR4
Type: String

address5
Defines the fifth line of the billing address for the company.

Table and Column: ARMASTER.ADDR5
Type: String

attentionName
Defines the name of the individual responsible for the company's billing.

Table and Column: ARMASTER.ATTENTION_NAME
Type: String

attentionPhone
Defines the phone number for the individual responsible for the company's billing.

Table and Column: ARMASTER.ATTENTION_PHONE
Type: String
**OBS Associations Schema Tag**

The OBS Associations tag is part of the schema mapping for the Company XOG object. No table is referenced for this tag. It is a wrapper for the OBSAssoc elements.

The OBS Associations tables include:
- **PRJ_OBS_ASSOCIATIONS**
- **PRJ_OBS_TYPES**
- **PRJ_OBS_UNIT**
- **PRJ_OBS_UNITS_FLAT**

This schema tag has the following attributes:

**completed**

Defines whether the OBS associations are complete. This field is optional. When completed, the value is True. Existing OBS associations not listed in the import are deleted.

**Table and Column:** Not applicable

**Type:** String

**Default:** False

**id**

Required. Defines the unique ID for the OBS type.

**Table and Column:** PRJ_OBS_TYPES.UNIQUE_NAME

**Type:** String

**name**

Defines the name of the OBS type.

**Table and Column:** PRJ_OBS_TYPES.NAME

**Type:** String

**unitPath**

Required. This is a slash-delimited list of unit names that lead to the unit with which the object is associated.

**Example:** "CAN/BC/VAN".

**Table and Column:** PRJ_OBS_UNITS.NAME

**Type:** String
Company Class

Use the Company Class XOG object to view inbound and outbound Company Class instances.

Schema Name

nikuxog_companyclass.xsd

Read and Write XML Files

The following XML files are included:

- companyClass_read.xml. Use this file to export Company Class instances from CA Clarity.
- companyClass_write.xml. Use this file to import Company Class instances that were previously exported from CA Clarity.

Prerequisites

None.

Read Filters

The following explicit read filters are used:

- **companyclass**
  
  Defines the company class name.

- **description**
  
  Defines the description for the Company Class.

- **shortdesc**
  
  Defines the short description for the Company Class.

Error Handling

The following errors can be thrown:

- Could not xog-in item because size of some attributes [description] is not within valid range.

Schema Mapping

Mappings for the following schema tag name is provided:

- **Company Class** (see page 162)
Company Class Schema Tag

The Company Class tag is part of the schema mapping for the Company Class XOG object. It has the following attributes:

**companyclass**
- Required. Defines the unique company class name.
- **Table and Column:** CLNTCLASS
- **Type:** String

**description**
- Required. Defines the description of the company class.
- **Table and Column:** DESCRIPTION
- **Type:** String

**shortdesc**
- Required. Defines the summary of the company class description.
- **Table and Column:** SHORTDESC
- **Type:** String
Content Pack

This object allows you to create new content for CA Clarity PPM. A content item is any item that displays in CA Clarity PPM but is not considered data. For example, a graph portlet is considered a content item, but a project is considered data.

Use this object to view inbound and outbound add-in items.

Schema Names

The following schema names are associated to the content pack XOG object:
- nikuxog_contentPack.xsd
- nikuxog_pageTypes.xsd
- nikuxog_filter_portlet.xsd

Read and Write XML Files

The following XML files are included:
- content_pack_read.xml. Use this file to export content packs from CA Clarity.
- content_pack_write.xml. Use this file to import content packs that were previously exported from CA Clarity.

Terms

The following terms are used when describing the content pack XOG object:

Content Item
Defines the user-defined add-in item that adds or extends CA Clarity PPM.

Queries
Defines the queries.

Reports and jobs
Defines the reports and jobs. You cannot import or export system-supplied reports using the XOG. They are not considered user-defined add-in items.

Portlets (graph, grid, and HTML)
Defines the portlets. You cannot import or export system-supplied portlets using the XOG. They are not considered user-defined add-in items.

Pages
Defines the pages.

Lookups
Defines the lookups. You can export system-supplied lookups.
Menus (menu manager sections and links)

Defines the menus.

Content Pack

Defines the collection of content that is bundled into a single distributable package.

Business Rules and Processing

The following business rules and processing apply to this XOG:

- When creating add-in items, it is recommended that you set up a staging server that is separate from the production server. After reviewing, testing, and verifying the effectiveness and usability of the new add-in items, you can determine that the add-in items are ready for production. You can then transfer the items to the production server. During this process, the items are exported from the staging server and imported to a production server.

- Instead of exporting and importing a series of add-in item XML (XOG) files, you can export many add-in items, then import them as a single XML file.

For example

You can:

1. Create a page that contains add-in items (portlets, queries, and lookups).
2. Add the portlets to a menu.
3. Run the XOG and export the content.
   
   The resulting XML file contains all of the add-in items. The file is fully annotated to describe how each element is filtered.

The attribute values for the filterMapping element (part of the PortletReferenceType complex type) has the following prerequisites unique to the filter portlet type:

- The filterPortletCode attribute value must match the portlet code of a valid filter portlet defined in the filter portlet schema.
- The fieldCode attribute value must match the field code of a valid filter portlet defined in the filter portlet schema.
- The dataProviderItemCode attribute value must match the column code of a valid grid or graph portlet column.
- The dataProviderItemCode and fieldCode attribute values must reference an attribute of the same data type.
- If the data type is a lookup, then the lookup type codes need to be the same between the attributes.

Dependencies
Add-in items have dependencies that are automatically resolved by the XOG. For example, if a query depends on a lookup and the query is imported, the lookup must also be imported (if it does not already exist). The following dependencies exist when exporting add-in items:

<table>
<thead>
<tr>
<th>Content Item Depends On</th>
<th>Content Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queries</td>
<td>Lookups</td>
</tr>
<tr>
<td>Graphs and Grids</td>
<td>Queries or Lookups</td>
</tr>
<tr>
<td>Reports and Jobs</td>
<td>Lookups</td>
</tr>
<tr>
<td>Pages, Portlets, and Queries</td>
<td>Lookups (sometimes Objects if data provider of portlet is object)</td>
</tr>
<tr>
<td>Menu Links</td>
<td>Pages</td>
</tr>
<tr>
<td>All items</td>
<td>Security</td>
</tr>
<tr>
<td>Objects</td>
<td>Lookups</td>
</tr>
</tbody>
</table>
OBS associations for pages, portlets, jobs and reports will be exported. They are also imported provided that the OBS and OBS unit (including parentage) exist in CA Clarity PPM. The same is true of security, which is imported if the user, group, or OBS unit exists.

**Import Rules**

Each add-in import request is regarded as a single transaction. If one item fails, the entire add-in is not imported. Add-in items are imported from XML files, which can be produced during the export or created manually.

If an item is modified to include new entries before an item is imported, the new entry is not affected by the import process; the import operation ignores the new entry.

**For example**

Consider a static lookup named SAMPLE_LOOKUP that contains the following values:

- OPEN
- CLOSED
- IN PROGRESS

You then add a new lookup value of SUSPENDED and change the existing label from CLOSED to FINISHED. Then import a Best Practice Accelerator that includes the definition for SAMPLE_LOOKUP which contains each of the three lookup values (OPEN, CLOSED, and IN PROGRESS) and the new lookup value (DELAYED). When the import operation completes, CA Clarity PPM contains the following lookup values:

- OPEN
- CLOSED (the change is overwritten)
- IN PROGRESS
- SUSPENDED (addition made to CA Clarity PPM using the application user interface is preserved)
- DELAYED (new item that was present in the input Best Practice Accelerator file is also added)

When you import an add-in item, it is updated with the definition that is present in the file. If a content item contains a new addition that does not exist in CA Clarity PPM, the new addition is also created in CA Clarity PPM. When importing an existing lookup, all rules described in the following table apply. The following table shows the various import rules associated with a lookup and the lookup values. These rules are honored by the user interface. As a result, the XOG import process for lookups honors these rules.

<table>
<thead>
<tr>
<th>Function</th>
<th>System-restricted</th>
<th>System</th>
<th>User-defined</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Static List Lookups</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>System-restricted</td>
<td>System</td>
<td>User-defined</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------</td>
<td>--------</td>
<td>--------------</td>
</tr>
<tr>
<td>Change Lookup Name and Description</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Change Sort Order (Manual or Alphanumeric)</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Deactivate or Activate Lookup</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Delete Lookup</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Change Lookup Value Name and Description</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Create New Lookup Values</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Deactivate or Activate Lookup Values</td>
<td>No</td>
<td>Yes (user-defined values)</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No (seeded values)</td>
<td></td>
</tr>
<tr>
<td>Reorder Lookup Values</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Dynamic Niku Query Lookups**

<table>
<thead>
<tr>
<th>Function</th>
<th>System-restricted</th>
<th>System</th>
<th>User-defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Lookup Name and Description</td>
<td>Yes</td>
<td>Does not apply</td>
<td>Yes</td>
</tr>
<tr>
<td>Edit Query</td>
<td>No</td>
<td>Does not apply</td>
<td>Yes</td>
</tr>
<tr>
<td>Edit Parent Window Fields</td>
<td>No</td>
<td>Does not apply</td>
<td>Yes</td>
</tr>
<tr>
<td>Edit Browse Window Name and Label Fields</td>
<td>Yes</td>
<td>Does not apply</td>
<td>Yes</td>
</tr>
<tr>
<td>Edit Browse Window Field Element Type</td>
<td>No</td>
<td>Does not apply</td>
<td>Yes</td>
</tr>
<tr>
<td>Edit Browse Window Selected Fields for Filter and List</td>
<td>No</td>
<td>Does not apply</td>
<td>Yes</td>
</tr>
<tr>
<td>Edit Browse Window Filter Field, List Column Order</td>
<td>Yes</td>
<td>Does not apply</td>
<td>Yes</td>
</tr>
<tr>
<td>Edit Browse Window Default Sort Column/Order</td>
<td>Yes</td>
<td>Does not apply</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Export Rules

You can export multiple add-in items at the same time. Add-in items can be filtered so that a subset of items can be exported. For example, a system may contain four HTML portlets. The export interface allows for two HTML portlets to be exported at the same time.

Portlets

This feature does not implement business rules for portlets. Most of the attributes and elements are either optional or have default values. While the XML schema does not validate such instances, the import mechanism does validate and generate an error for invalid grids or graphs.

For example, if a Grid portlet is based on a query that does not contain metrics, then no `<metricColumn>` elements should be present in the imported XML schema.

Jobs and reports definitions

During import, the imported attributes of the definition match the user interface exactly. When parameters are changed, the old parameters are deleted and new ones are added. Existing scheduled jobs and reports are cancelled, since the parameters no longer exist. All saved parameters are deleted. If the update flag is set to "True", then only captions (job and parameters) can be changed. If the update flag is set to "False", the content item is assumed to contain all the parameters and existing parameters are deleted as described above.

Pages

The export of a page includes all non-system portlets. OBS and security associations can be specified and, as with other items, a Warning is written to the log file if the target principal (OBS unit, Group, or User) does not exist.

Lookups

Lookups are either static or dynamic: Dynamic lookups are based on NSQL, Static lookups do not contain the NSQL values contained in dynamic lookups and are generated by executing the NSQL at runtime when they are rendered on a page as a pull-down or as a browse page. Static lookup values are seeded and persist as individual rows in CA Clarity (in the table named CMN_LOOKUP).

While it is true that for any content item that is part of a Best Practice Accelerator, system objects are not imported or exported, there are exceptions with regard to lookups. Lookups are classified into three categories: System-restricted, System, and User-defined. All lookup types (dynamic and static) can be exported.

Queries

This feature follows all business rules for the query user interface. Items that can be updated via the user interface can be updated with an import request. Although the queries schema allows for multiple, vendor-specific NSQL text, only valid text is imported into CA Clarity. For example, an import request on an Oracle system that contains both `dbVendor="mssql"` and `dbVendor="oracle"` imports only the `dbVendor="oracle"`. 
Text with dbVendor="all" is always imported.

When creating queries, the user in the Administration Tool always sets dbVendor="all". This value cannot be changed in the user interface. Should a query be vendor-specific, you can set the dbVendor attribute value in the XML document.

When querying for items to create a Best Practice Accelerator, you must explicitly query for each type of item in the export (read) request. The following statement returns all portlets that include source="acme.com". Without the <PortletQuery> element in the read request, no portlets are imported. For example:

```xml
<PortletQuery>
  <Filter name="source">
    criteria="EQUALS">acme.com</Filter>
  </PortletQuery>
```

All add-in items have a source attribute which is used to differentiate originating-system items from CA Clarity PPM-supplied items. The source attribute maps to the column source in various database tables. To change the source for add-in items created with the user interface, use SQL to change the defaultValue for this column; then new add-in items will have the source specified.

This applies to the following tables:

- CMN_PORTLETS
- CMN_PAGES
- CMN_SCH_JOB_DEFINITIONS
- CMN_LOOKUP_TYPES
- CMN_GG_NSQ_QUERY

The SQL to change this is:

```
ALTER TABLE CMN_LOOKUP_TYPES MODIFY SOURCE VARCHAR (80) DEFAULT 'acme.com'
```

**Error Handling**

PageTypes filter mappings that do not meet the dependencies (described earlier), will not throw an error, but the mapping will not be implemented.

**Schema Mapping**

The following schema tags and attributes define the ContentPack filter portlet schema (nikuxog_pageTypes.xsd):

- filterMapping
- portlet
Filter Mapping (filterMapping) Schema Tag

The filter mapping tag is part of the schema mapping for the Content Pack XOG object. The following attributes describe the mapping of a field on a filter portlet to a data provider attribute of a grid or graph portlet. This element is only valid as a child of a grid or graph portlet reference.

The filterMapping schema tag has the following attributes:

- **filterPortletCode**
  - Required. Defines the code of the filter portlet.
  - **Table and Column:** Portlet ID
  - **Type:** String

- **fieldCode**
  - Required. Defines the code of the filter field.
  - **Table and Column:** Field ID
  - **Type:** String

- **dataProviderItemCode**
  - Required. The code of the data provider property or metric.
  - **Table and Column:** Attribute ID
  - **Type:** String

- **hidePortletWhenEmpty**
  - When true, the referenced grid or graph portlet is hidden when no filter value is present.
  - **Table and Column:** Hide If Empty
  - **Type:** Boolean

portlet Schema Tag

This tag is part of the schema mapping for the Content Pack XOG object. The following schema tags and attributes define the ContentPack filter portlet schemas (nikuxog_contentPack.xsd and nikuxog_filter_portlet.xsd):

- **filterPortlet**
- **Field**
- **lookupParam**

The Portlet schema tag has the following attributes:
**defaultFilter**

The default filter for the page. It can be true for only one portlet on the page.

*Table and Column:* Default

*Type:* Boolean

**pageLevelFilter**

Defines whether the filter values persist across the page.

*Values:*

- True. The filter values persist across this page only.
- False. The filter values apply throughout CA Clarity.

*Table and Column:* Persist

*Type:* Boolean

**filterPortlet Schema Tag**

The filterPortlet schema tag inherits existing PortletAttributesTypes found in nikuxog_portlet.xsd and NLS, OBSAssocs, Security, and FilterViewTypes. These items are consistent across all portlets and not specific to filterPortlet. They are not explicitly listed.

The filterPortlet schema tag has the following attributes:

**uiType**

Defines the UI rendering type.

*Table and Column:* Render As

*Type:* FilterPortlet UI Type (section, toolbar)

**isCollapsed**

Defines the default section state.

*Table and Column:* Default Filter State

*Type:* Boolean

**bgColorKey**

The session key that controls the background color of the toolbar filter portlets.

*Table and Column:* None

*Type:* String
Field Schema Tag

The Field schema tag is a filter field, and has the following attributes:

- **nls**
  - Required. The name and description of the field.
  - **Table and Column:** Field Name/Description
  - **Type:** NlsType

- **tip**
  - Defines the tooltip
  - **Table and Column:** Tooltip
  - **Type:** NlsType

- **hint**
  - Defines the hint or instructional text.
  - **Table and Column:** Hint
  - **Type:** NlsType

- **defaultValue**
  - Defines the default value for the field.
  - **Table and Column:** Filter Default
  - **Type:** Any type

- **lookupParam**
  - Defines a parameter of a parameterized lookup.
  - **Table and Column:** Lookup Parameter Mappings
  - **Type:** See the lookupParam table

- **code**
  - Required. Defines the code of the field.
  - **Table and Column:** Field Id
  - **Type:** String

- **dataType**
  - Required. Defines the data type of the field.
  - **Table and Column:** Data Type
  - **Type:** String
**extDataType**

Defines the extended type of the field.

**Table and Column:** N/A

**Type:** String

**widgetType**

Required. Defines the display type of the field.

**Table and Column:** Display Type

**Type:** String

**extWidgetType**

Required. The extended display type of the field.

**Table and Column:** N/A

**Type:** String

**lookupTypeCode**

Defines the lookup code for a lookup field.

**Table and Column:** Lookup

**Type:** String

**width**

Defines the width of the filter field.

**Table and Column:** Width

**Type:** Integer

**position**

Defines the position of the filter field.

**Table and Column:** Layout

**Type:** Integer

**multiValued**

Defines the lookup style of a lookup field.

**Table and Column:** Lookup Style

**Type:** Boolean

**multiValuedLookup**

Indicates if the lookup field is a Multi Valued Lookup data type.

**Table and Column:** N/A

**Type:** Boolean
fixedWidget
Indicates if the field is fixed.
Table and Column: N/A
Type: Boolean

hidden
Indicates if the field is hidden.
Table and Column: Hidden in filter
Type: Boolean

required
Indicates if the field requires a value.
Table and Column: Required in Filter
Type: Boolean

readOnly
Indicates if the field’s value is read only.
Table and Column: Read-Only in Filter
Type: Boolean

minValue
Defines the minimum default value of a date or numeric range field.
Table and Column: Filter Default From
Type: String

maxValue
Defines the maximum default value of a date or numeric range field.
Table and Column: Filter Default To
Type: String

columnNumber
Defines the field’s column or row in the corresponding section or toolbar filter.
Table and Column: Layout
Type: Integer

depParentLookupTypeCode
Defines the lookup code of a dependent lookup.
Table and Column: Lookup
Type: String
**depEntry**

Defines the lookup code for entry into dependent lookup.

*Table and Column:* Entry  
*Type:* String

**depExit**

Defines the lookup code for exit into dependent lookup.

*Table and Column:* Exit  
*Type:* String

### lookupParam Schema Tag

The `lookupParam` schema tag is a parameter of a parameterized lookup, and has the following attributes:

**code**

Required. Defines the field code of the parameter.

*Table and Column:* Field Code  
*Type:* String

**dpCode**

Required. Defines the NSQL parameter specified in the lookup.

*Table and Column:* Lookup Parameter  
*Type:* String
Cost Plan

A cost plan is created for an investment that already exists. Use the Cost Plan XOG object to view inbound and outbound financial cost plans.

Schema Name

nikuxog_costPlan.xsd

Read and Write XML Files

The following XML files are included:
- costPlan_read.xml. Use this file to export cost plans from CA Clarity.
- costPlan_write.xml. Use this file to import cost plans that were previously exported from CA Clarity.

Prerequisites

Before using this XOG, make sure the following objects already exist in CA Clarity:
- Investments
- Entity
- Time periods
- Grouping attributes

Business Rules and Processing

The following business rules and processing apply to this XOG:
- A Cost Plan object is created by configuring the cost plan setup properties.
- Cost plan details (line items) are added to the plan.
- Existing plan detail records are not deleted.

Read Filters

The following explicit read filters are used:
- code. The code for the cost plan.
- name. The name of the cost plan.
- investmentCode. The investment code with which the plan is associated.

Error Handling

When importing or exporting the cost plan, the following errors can be thrown:
- You must have Plan XOG access rights to perform this action.
- Plan code is required.
- Grouping Attribute is missing or invalid.
- Investment Code is missing or invalid.
- Period Type cannot be changed once defined.
- Grouping Attributes cannot be changed once defined.
- Investment must be associated to an Entity before setting up financial plans.
- Benefit plan ID is missing or invalid.
- Period Date is missing or invalid.
- Grouping Attributes in Plan Details do not match Plan Grouping Attributes.
- GL Account is missing or invalid.
- Missing or invalid value for Grouping Attribute.
- Finish time period is invalid.
- Start time period is invalid.
- Grouping Attributes do not match locked plan structure for associated entity.
- Cannot modify plan periods prior to the freeze date.
- Valid Fiscal Period does not exist for start and end dates for plan detail.

**Schema Mappings**

The following schema tag names are described:
- [Cost Plan](see page 177)
- [Description](see page 179)
- [Detail](see page 180)
- [Grouping Attributes](see page 179)
- [Segment](see page 180)

**CostPlan Schema Tag**

The Cost Plan tag is part of the schema mapping for the Cost Plan XOG object. This tag has the following attributes:

- **benefitPlanCode**
  - Defines the ID of the benefit plan that is associated to the cost plan.
  - **Table and Column:** BENEFIT_PLAN_ID
  - **Type:** String
Cost Plan

Code
Required. Defines the unique ID of the cost plan for XOG.

*Table and Column:* CODE
*Type:* String

FinishPeriod
Required. Defines the finish time period name.

*Table and Column:* END_PERIOD_ID
*Type:* String

Investment Code
Required. Defines the investment code.

*Table and Column:* OBJECT_ID
*Type:* String

Investment Type
Defines the investment type. This is used only in the XOG read result.

*Type:* String

isPlanOfRecord
Indicates if the cost plan is the plan of record. If this is not set and it is the first plan of the investment, then the plan is marked as the plan of record.

*Table and Column:* IS_PLAN_OF_RECORD
*Type:* String

Name
Required. Defines the name of the cost plan.

*Table and Column:* NAME
*Type:* String

Period Type
Required. Defines the time period type.

*Table and Column:* PERIOD_TYPE_CODE
*Type:* String

Revision
This is used only in XOG read result. It represents the revision of the cost plan.

*Table and Column:* REVISION
*Type:* Integer
**StartPeriod**

Required. Defines the start time period name.

**Table and Column:** START_PERIOD_ID

**Type:** String

---

**Description Schema Tag**

This tag is part of the schema mapping for the Cost Plan XOG object. This schema tag is a text node.

**Description**

Defines the cost plan description.

**Table and Column:** description

**Type:** String

---

**Grouping Attributes Schema Tag**

This tag is part of the schema mapping for the Cost Plan XOG object. This tag is used to define grouping attributes for a cost plan. The tag includes the following attribute:

**Grouping Attribute**

Specifies grouping attributes for a cost plan.

**Possible Values:**

- charge_code_id
- role_id
- resource_id
- department_id
- location_id
- transaction_class_id
- resource_class_id
- input_type_code_id
- lov1_id
- lov2_id

**Table and Column:** The value is stored in the ODF_MULTI_VALUED_ATTRIBUTES table.

**Type:** String
**Detail Schema Tag**

This tag is part of the schema mapping for the Cost Plan XOG object. This tag defines a cost plan detail row and is composed primarily of schema tags.

**Note:** The Detail schema tag also has two attributes: glAccountMain and glAccountSub. These attributes define the GL Account associated with the Detail schema tag. The attributes map to the GL_ACCOUNT_ID column.

The Detail schema tag includes the following tags:

**Cost**

Defines the cost for specific time period segments.

**Table and Column:** COST

**Units**

Defines the quantity for a specific time period segment.

**Table and Column:** UNITS

**Revenue**

Defines the revenue for a specific time period segment.

**Table and Column:** REVENUE

**Grouping Attributes**

Defines the grouping attribute codes and values for the grouping attributes selected for the cost plan. Each GroupingAttribute represents a code and value pair for an attribute.

**Table and Column:** The value is stored in the column corresponding to the groupingAttribute code (for example, location_id, charge_code_id).

**Type:** String

**Custom Information**

Defines the field names for the custom information.

---

**Segment Schema Tag**

This tag is part of the schema mapping for the Cost Plan XOG object. The Unit, Cost, and Revenue schema tags will include one or more Segment schema tags. This tag has the following attributes:

**start**

Defines the start period for the unit, cost, or revenue.

**Type:** String
finish

Defines the end period for the unit, cost, or revenue.

Type: String

value

Defines the value for the unit, cost, or revenue.

Type: String
Cost Plus Code

Use the cost plus code XOG object to view inbound and outbound cost plus code object instances. Cost Plus Codes are an alternate method of determining rates by adding markups to either the Standard or Actual cost.

Schema Name

nikuxog_costPlusCode.xsd

Read and Write XML Files

The following XML files are included:
- costPlusCodes_read.xml. Use this file to export cost plus codes from CA Clarity.
- costPlusCodes_write.xml. Use this file to import cost plus codes that were previously exported from CA Clarity.

Prerequisites

None

Read Filters

The following explicit read filters are used:
- code
  Defines the code for the cost plus code.
- description
  Defines the description for the cost plus code.
- appliesTo
  Defines the apply to Actual/Standard parameter (values are StandardCost or ActualCost).

Error Handling

When importing or exporting the cost plan, the following error can be thrown:
Failed to export cost plus codes.

Schema Mapping

Mappings for the following schema tag name is provided:
- Cost Plus Code (see page 183)
Cost Plus Rule (costplusrule) Schema Tag

The cost plus rule tag is part schema mapping for the Cost Plus Code XOG object. It has the following attributes:

unitsFrom
   Required. Defines the beginning units of the cost plus rule.
   Table and Column: COSTPLUSRULES.FROMRANGE
   Type: Date

unitsTo
   Required. Defines the end units of the cost plus rule.
   Table and Column: COSTPLUSRULES.TORANGE
   Type: Date

multiplierAmount
   Defines the multiplier amount for the cost plus rule.
   Table and Column: COSTPLUSRULES.MULTAMOUNT
   Type: Double

burdenAmount
   Defines the burden amount for the cost plus rule.
   Table and Column: COSTPLUSRULES.BURDENAMOUNT
   Type: Double

overheadAmount
   Defines the overhead amount for the cost plus rule.
   Table and Column: COSTPLUSRULES.OVERHEADAMOUNT
   Type: Double
Department

Use the department XOG object to view inbound and outbound department attributes.

Schema Name

nikuxog_department.xsd

Read and Write XML Files

The following XML files are included:
- department_read.xml. Use this file to export departments from CA Clarity.
- department_write.xml. Use this file to import departments that were previously exported from CA Clarity.

Prerequisites

Before using this XOG, you must ensure that an entity exists.

Business Rules and Processing

When a department is created, a corresponding OBS unit is created in the department OBS referred to by the department's entity.

Read Filters

The following explicit read filter is used:

Entity

The unique entity code for which the departments should be read out.

Error Handling (Writes & Updates)

Errors are thrown based on the following checks:
- Entity: Checks if the entity is valid and exists.
- Required fields: Ensures all required fields have values.
- Location associations: Ensures locations belong to the same entity. If the location does not exist, a warning is output.

Schema Mappings

The following schema tag names are provided to XOG departments:
- Departments
- Description
- LocationAssociations
- Budget
- Child Department

**Departments Schema Tag**

This tag is part of the schema mapping for the Department XOG object. This is a placeholder tag for multiple departments.

**Department**

The actual department object. Department has the following attributes:

- **department_code**
  - Required. Defines the unique department code.
  - **Table and Column:** DEPARTMENTS.departcode
  - **Type:** String

- **short_description**
  - Required. Defines the department name.
  - **Table and Column:** DEPARTMENTS.shortdesc
  - **Type:** String

- **dept_identifier**
  - Defines the general ledger segment value mapped to this department.
  - **Table and Column:** DEPARTMENTS.departidentifier
  - **Type:** String

- **default_reviewer**
  - Defines the default reviewer for the department.
  - **Table and Column:** DEPARTMENTS.default_reviewer
  - **Type:** String

- **alt_default_reviewer**
  - defines the alternate reviewer for the department.
  - **Table and Column:** DEPARTMENTS.alt_default_reviewer
  - **Type:** String
**parent_department_code**

Defines the code for parent department.

**Table and Column:** parent_department_id

**Type:** String

**dept_manager_code**

The department manager resource code.

**Table and Column:** DEPARTMENTS.department_manager_id

**Type:** String

**brm_code**

Defines the business relationship manager.

**Table and Column:** brm_id

**Type:** String

**entity**

Required. The identify for the entity to which the department belongs.

**Table and Column:** DEPARTMENTS.entity_id

**Type:** String

---

**Description Schema Tag**

This tag is part of the schema mapping for the Department XOG object. It is used for the department description. This tag has the following attribute.

**Description**

Required. Defines the description tag.

**Table and Column:** description

**Type:** String
LocationAssociations Schema Tag

This tag is part of the schema mapping for the department XOG object. The placeholder tag for multiple location associations.

LocationAssociation

A location associated to a department. This has the following attribute.

locationcode

Required. Defines the location code. Location must belong to the same entity as the department.

Table and Column: locn_id
Type: String

Budget Schema Tag

The budget tag is part of the schema mapping for the department XOG object. A simple budget including the project’s planned cost, NPV, ROI, and breakeven information. The values apply to only one time period from the start date to the finish date of the project.

Subscriptions Schema Tag

The subscriptions tag is part of the schema mapping for the department XOG object.

It is a placeholder tag for the services to which the department subscribes.

Subscription

The service to which the department subscribes and its properties. This has the following attributes.

sla_violations

Defines the number of SLA violations.

Table and Column: DPT_SUBSCRIPTIONS.sla_violations
Type: Integer

sla_violations_th

Defines the threshold for SLA violations.

Table and Column: DPT_SUBSCRIPTIONS.sla_violations_threshold
Type: Integer
incidents
Defines the number of incidents.
**Table and Column:** DPT_SUBSCRIPTIONS.incidents
**Type:** Integer

incidents_threshold
Defines the threshold for incidents.
**Table and Column:** DPT_SUBSCRIPTIONS.incidents_threshold
**Type:** Integer

change_orders
Defines the number of change orders.
**Table and Column:** DPT_SUBSCRIPTIONS.change_orders
**Type:** Integer

charges
Defines the total charges (from chargebacks) against the investment (service) for this subscription.
**Table and Column:** DPT_SUBSCRIPTIONS.charges
**Type:** Integer

cust_satisfaction
Defines the customer satisfaction rating for this subscription.
**Table and Column:** DPT_SUBSCRIPTIONS.customer_satisfaction
**Type:** Integer

total_users
Defines the total number of users utilizing this subscription.
**Table and Column:** DPT_SUBSCRIPTIONS.total_users
**Type:** Integer

active_users
Defines the number of active users utilizing this subscription.
**Table and Column:** DPT_SUBSCRIPTIONS.active_users
**Type:** Integer

page_hits
Defines the page hits as captured for this subscription if applicable.
**Table and Column:** DPT_SUBSCRIPTIONS.page_hits
**Type:** Integer
**entityId**

Required. Defines the entity to which the service belongs.

**Table and Column**: This is a derived attribute.

**Type**: String

**departmentId**

Required. Identifies the subscribing department.

**Table and Column**: DPT_SUBSCRIPTIONS.department_id

**Type**: String

**serviceId**

Required. Defines the identifier that makes it unique in combination with the table_name column.

**Column**: pk_id

**Type**: String

**Department (Child Department) Schema Tag**

This tag is part of the schema mapping for the Department XOG object. A child department has all the elements and attributes of the parent department.

**obsTypes Schema Tag**

This tag is part of the schema mapping for the Department XOG object. It is a placeholder for the two OBS types that represent the Location and Department OBS's. This is similar to the generic OBS XOG structure.

This schema tag is composed of the following:

- obs
- level
- obsAssociations
- obs

The obsTypes schema tag has the following attributes:

**code**

Required. Defines the unique code for OBS type.

**Table and Column**: PRJ_OBS_TYPES.unique_name

**Type**: String
name
   Required. Defines the name of the OBS type.
   **Table and Column:** PRJ_OBS_TYPES.name
   **Type:** String

description
   Required. Describes the OBS type.
   **Table and Column:** PRJ_OBS_TYPES.description
   **Type:** String

level
   Level represents a level in the OBS type. At least one level is required. Level has the following attributes.
   
   Name
      Required. The name for the level.
      **Table and Column:** Prj_Obs_Levels.Name
      **Type:** String

   depth
      Required. The depth of the OBS level.
      **Table and Column:** Prj_Obs_Levels.obs_level
      **Type:** Integer

obsAssociation
   The object types that are associated to this OBS type. This tag is optional.

   object
      Required. The name of the object type associated to this OBS.
      **Table and Name:** Prj_Obs_Associations.table_name
      **Type:** String

   associationType
      Required.
      **Table and Name:** Prj_Obs_Associations.is_leaf_only
      **Type:** String
Entity

Use the entity XOG object to view inbound and outbound entity attributes.

Schema Name

nikuxog_entity.xsd

Read and Write XML Files

The following XML files are included:

- entity_read.xml. Use this file to export entities from CA Clarity.
- entity_write.xml. Use this file to import entities that were previously exported from CA Clarity.

Prerequisites

None.

Business Rules and Processing

The following business rules and processing apply to this XOG object:

- The entity inherits the currency type from the system settings.
- In entity create mode, if the OBS types mentioned in the XOG file do not exist, the OBS types are created automatically.
- Structural updates to OBS types are not allowed if they are referred to by the entities. Use the location and department XOG files for structural changes.
- Ensures that the OBS types are not referred to by another entity.
- Creates departments and locations for the OBS units (when in create mode).

Read Filters

The following explicit read filter is used:

Entity

Defines the unique entity code that needs to be read out.

Description

Defines the description of the entity.

Schema Mappings

The following schema tag attributes are described:

- Entity
Entity Schema Tag

This tag is part of the schema mapping for the Entity XOG object. It is a placeholder tag for multiple entities.

Entity

The actual entity object. The entity schema mapping includes definitions for the home and reporting currencies. This schema tag has the following attributes:

entity

Required. Defines the name of the entity. Do not allow truncation.

Table and Column: ENTITY.entity
Type: String

finPeriodCode

Optional. Defines the fiscal time period type of the entity.

Possible Values: WEEKLY, 13_PERIODS_PER_YEAR, SEMI_MONTHLY, MONTHLY, QUARTERLY and ANNUALLY.

Table and Column: ENTITY.FIN_PERIOD_CODE
Type: String

geoOBS

Required. Refers to the OBS that will represent the geographical structure (that is, locations).

Table and Column: ENTITY.geo_chart_obs_type_id
Type: String

orgOBS

Required. Refers to the OBS that will represent the organizational structure (that is, departments).

Table and Column: ENTITY.org_chart_obs_type_id
Type: String
homeCurrency

Required. Defines the lookup values for each ISO standard code. You must validate that it is an active currency.

**Table and Column:** ENTITY.Home_Currency_Code
**Type:** String

reportingCurrency

Required. Defines the lookup values for each ISO standard code. You must validate that it is an active currency.

**Table and Column:** ENTITY.Reporting_Currency_Code
**Type:** String

billingCurrency

Optional. Defines the lookup values for each ISO standard code. You must validate that the billing currency is an active currency.

**Table and Column:** ENTITY.Billing_Currency_Code
**Type:** String

externalID

Required. Refers to the originating system’s set of books ID.

**Table and Column:** ENTITY.External_ID
**Type:** String

defaultProjectClass

Represents the project class for the entity. Validated against project classes.

**Table and Column:** ENTITY.PROJECT_CLASS
**Type:** String

defaultWIPClass

Represents the default WIP class for the entity. Validated against WIP classes

**Table and Column:** ENTITY.WIP_CLASS
**Type:** String

defaultClientClass

Represents the default client class for the entity.

**Table and Column:** ENTITY_CLIENT_CLASS
**Type:** String
defaultBatchCycle

Represents the default batch cycle for the entity.

**Table and Column:** ?

**Type:** String

defaultLaborRateSource

Defines the default rate matrix for labor transactions for the entity. Validated against matrices.

**Table and Column:** ENTITY.TRANS_RATE_SOURCE_LABOR

**Type:** String (in schema). Number (in CA Clarity).

defaultLaborCostSource

Defines the default cost rate matrix for labor transactions for the entity. Validated against matrices.

**Table and Column:** ENTITY.TRANS_COST_SOURCE_LABOR

**Type:** String (in schema). Number (in CA Clarity).

defaultLaborSourceLocation

Defines the default location for labor transactions for the entity.

**Table and Column:** ENTITY.TRANS_LOCATION_LABOR

**Type:** String

Values: Project and Resource

defaultMaterialRateSource

Optional. Defines the default rate matrix for rate of material transactions for the entity. Validated against matrices.

**Table and Column:** ENTITY.TRANS_RATE_SOURCE_MATERIALS

**Type:** String (in schema). Number (in CA Clarity).

defaultEquipmentRateSource

Optional. Defines the default rate matrix ID for equipment transactions for the entity. Validated against matrices.

**Table and Column:** ENTITY.TRANS_RATE_SOURCE_EQUIPMENT

**Type:** String (in schema). Number (in CA Clarity).

defaultExpenseRateSource

Optional. Defines the default rate matrix for expense transactions for the entity. Validated against matrices.

**Table and Column:** ENTITY.TRANS_RATE_SOURCE_EXPENSE

**Type:** String (in schema). Number (in CA Clarity).
exchangeRateTypeLabor
Optional. Defines the labor exchange rate type for the entity.

**Table and Column:** ENTITY. EXCHANGE_RATE_TYPE_LABOR

**Type:** String

exchangeRateTypeMaterials
Optional. Defines the materials exchange rate type for the entity.

**Table and Column:** ENTITY. EXCHANGE_RATE_TYPE_MATERIALS

**Type:** String

exchangeRateTypeEquipment
Optional. Defines the equipment exchange rate type for the entity.

**Table and Column:** ENTITY. EXCHANGE_RATE_TYPE_EQUIPMENT

**Type:** String

exchangeRateTypeExpense
Optional. Defines the expense exchange rate type for entity.

**Table and Column:** ENTITY. EXCHANGE_RATE_TYPE_EXPENSE

**Type:** String

**Description Schema Tag**

This tag is part of the schema mapping for the Entity XOG object. It is used to describe the entity. It has the following attribute:

**Description**

Required. Defines the description of the entity.

**Table and Column:** ENTITY.description

**Type:** String

**Short Description (shortDescription) Schema Tag**

This tag is part of the schema mapping for the Entity XOG object. The short description for the entity. It has the following attribute:

**short description**

Required. Defines the short description tag.

**Table and Column:** ENTITY.shortdesc

**Type:** String
OBS Types (obsTypes) Schema Tag

This tag is part of the schema mapping for the Entity XOG object. The tag describes the OBS types associated with the entity.

This obsTypes schema tag can include 0 to n number of obs tags.

Each obs tag is composed of the following child elements:
- **level** (minimum 1, maximum 10)
- **objectAssociation** (minimum 0, maximum unbounded)
- **unit** (minimum 0, maximum unbounded)

**obs**

The obs tag has the following attributes:

- **code**
  - Required. Defines the unique code for OBS type.
  - **Table and Column**: `PRJ_OBS_TYPES.unique_name`
  - **Type**: String

- **name**
  - Required. Defines the name of the OBS type.
  - **Table and Column**: `PRJ_OBS_TYPES.name`
  - **Type**: String

- **description**
  - Optional. Describes the OBS type.
  - **Table and Column**: `PRJ_OBS_TYPES.description`
  - **Type**: String

**level**

This child element represents a level in the OBS type. At least one level is required. **level** has the following attributes:

- **Name**
  - Required. The name for the level.
  - **Table and Column**: `Prj_Obs_Levels.Name`
  - **Type**: String
**depth**

Required. The depth of the OBS level.

*Table and Column:* Prj_Obs_Levels.obs_level

*Type:* Integer (allowed value 1 to 10)

---

**objectAssociation**

This child element represents the object types associated with the OBS type. This tag is optional.

**object**

Required. The name of the object type associated to this OBS.

*Table and Name:* Pri_Obs_Associations.table_name

*Type:* String

**associationType**

Required.

*Possible Values:* Any Unit, Lowest Level

*Table and Name:* prj_obs_object_types.is_leaf_only

*Type:* String

---

**unit**

This unit tag represents the units for this OBS type. This tag is optional.

Each unit is composed of the following child elements:

- associatedObject
- rights
- Security

The unit schema tag has the following attributes:

**code**

Required. Defines the unique code for OBS unit.

*Table and Column:* PRJ_OBS_UNITS.unique_name

*Type:* String

**name**

Required. Defines the name of the OBS unit.

*Table and Column:* PRJ_OBS_UNITS.name

*Type:* String
**GL Periods [GLPeriods] Schema tag**

This tag is part of the schema mapping for the Entity XOG object. The tag describes the GL periods associated with the entity. This schema tag has the following attributes:

- **periodName**
  - Required. Defines the GL period name.
  - **Table and Column**: ?
  - **Type**: String

- **period**
  - Required. Defines the GL period.
  - **Table and Column**: ?
  - **Type**: Non-negative integer

- **quarter**
  - Defines the GL quarter.
  - **Table and Column**: ?
  - **Type**: Non-negative integer

- **year**
  - Defines the GL year.
  - **Type**: Non-negative integer

- **startDate**
  - Required. Defines the GL period start date.
  - **Type**: String

- **endDate**
  - Required. Defines the GL period end date.
  - **Type**: String

- **description**
  - Optional. Describes the GL period.
  - **Type**: String

- **periodType**
  - Required. Describes the GL period type.
  - **Table and Column**: BIZ_COM_PERIODS.PERIOD_TYPE
  - **Type**: String
entityId

Required. Provides a unique ID for the GL period.

**Table and Column**: BIZ_COM_PERIODS.ENTITY_ID

**Type**: String (Number in database)

isActive

Optional. Indicates if the GL period is active

**Table and Column**: BIZ_COM_PERIODS.IS_ACTIVE

**Type**: Boolean (Number in database)

**Plan Defaults [PlanDefaults] Schema Tag**

This tag is part of the schema mapping for the Entity XOG object. The tag describes the plan defaults associated with the entity. An entity can have only one planDefaults.

This tag includes the child element GroupingAttributes.

The planDefaults schema tag includes the following attributes:

**periodTypeCode**

The fiscal time period type for planDefaults.

**Table and Column**: FIN_PLAN_DEFAULTS.PERIOD_TYPE_CODE

**Possible Values**: WEEKLY, 13_PERIODS_PER_YEAR, SEMI_MONTHLY, MONTHLY, QUARTERLY, and ANNUALLY.

**Type**: String

**startPeriodName**

The start period name for planDefaults.

**Table and Column**: FIN_PLAN_DEFAULTS.START_PERIOD_ID

**Type**: String (Number in database)

**endPeriodName**

The end period name for planDefaults.

**Table and Column**: FIN_PLAN_DEFAULTS.END_PERIOD_ID

**Type**: String (Number in database)
lockPlanStructure

Flag to identify whether to lock the plan structure (grouping attributes) for planDefaults.

**Table and Column:** FIN_PLAN_DEFAULTS. LOCK_PLAN_STRUCTURE
**Type:** Boolean (Number in database)

freezeDate

The freeze date for planDefaults. This means details pertaining to the financial plans of this entity can be edited only if the startDate of a particular period is after this freeze date.

**Table and Column:** FIN_PLAN_DEFAULTS. FREEZE_DATE
**Type:** Date

**Grouping Attributes Schema Tag**

This tag is a child element of the Plan Defaults schema tag. The tag can have a minimum of one and up to 10 grouping attributes.

The tag has the following attributes:

**Grouping Attribute**

Specifies the grouping attributes for the plan defaults for an entity.

**Table and Column:** ODF_MULTI_VALUED_LOOKUPS.VALUE
**Type:** String
Financial Transaction

Use the financial transaction XOG object to view inbound and outbound financial transaction attributes for investments. This XOG object exports WIP transactions from PPA_WWP table so that the data can be imported into an ERP or other enterprise system.

Schema Name

nikuxog_commonTransaction.xsd

Read and Write XML Files

The following XML files are included:

- imp_transactions_read.xml. Use this file to export financial transaction attributes for investments from CA Clarity.
- imp_transactions_write.xml. Use this file to import financial transaction attributes for investments that were previously exported from CA Clarity.

Prerequisites

Before using this XOG object, make sure that all foreign key references of the financial transaction (for example, resource, investment, and others) have been defined.

Business Rules and Processing

The following business rules and processing apply to this XOG object:

- The XOG processes WIP transactions based on the following fields:
  - **transactionType**
    - Defines the financial transaction type.
    - **Values:** L, M, X, or Q.
  - **projectID**
    - The valid project ID.
  - **clientID**
    - The valid company ID.
  - **transactionDate**
    - Enter Start Date, and End Date to get transaction data for all projects between the dates specified.
  - And and Or processing is supported among these fields.
  - Transactions on hold or in error are not processed.
Once a transaction is exported, XOG_EXPORTED is flagged and EXPORTED_DATE is stamped with the time and date in the PPA_WIP and PPA_BILLINGS tables. For adjustments and reversals, a negative amount is exported (or a positive amount if the posting was negative).

Read Filters

None

Schema Mappings

The schema mappings are provided for the following Financial Transaction tag names:

- Transactions (see page 202)
- Transaction Import (see page 207)

Transactions Schema Tag

This tag is part of the schema mapping for the Financial Transaction XOG object. It has the following attributes:

**transactionID**

Required. The transaction identifier. It must be unique.

*Table and Column:* TRANSNO

*Type:* Number

**applyToTransactionID**

Optional. If the transaction is an adjusted, reversed, or transferred transaction, then applyToTransactionID refers to the parent transaction. Otherwise this field is not public.

*Table and Column:* APPLYTO

*Type:* Number

**clientID**

Required. The company identifier.

*Table and Column:* COMPANY_CODE

*Type:* String

**clientName**

Required. Derived from company ID.

*Table and Column:* CLNTSUPP.COMPANY_NAME

*Type:* String
projectID
   Required. The project identifier.
   Table and Column: PROJECT_CODE
   Type: String

projectName
   Required. Derived from project ID
   Table and Column: SRM_PROJECTS.NAME
   Type: String

taskID
   Optional. A valid task identifier.
   Table and Column: TASK_ID
   Type: Number

taskName
   Optional. Derived from task ID.
   Table and Column: PRTASK. PRNAME
   Type: String

transactionDate
   Required. The date of the transaction.
   Table and Column: TRANSDATE
   Type: Date

resourceID
   Required. The resource identifier.
   Table and Column: RESOURCE_CODE
   Type: String

resourceName
   Required. Derived from resource ID.
   Table and Column: SRM_RESOURCES.FULL_NAME
   Type: String

roleID
   Optional. The role identifier.
   Table and Column: ROLE_CODE
   Type: String
**transactionType**
Required. The transaction type.
*Table and Column:* TRANSTYPE
Type: String

**chargeCode**
Required. A valid charge code.
*Table and Column:* COSTCODE
Type: String

**inputTypeCode**
Optional. Valid input type code
*Table and Column:* INPUT_TYPE
Type: String

**chargeable**
Required. Indicates if the transaction can be charged to the company.
*Values:*
- 0. Not chargeable
- 1. chargeable
*Table and Column:* CHARGEABLE
Type: Number

**units**
Required. The number of units.
*Table and Column:* QUANTITY
Type: Number

**CurrencyValue ->actualCostRate**
Required.
*Table and Column:* PPA_WIP_VALUES.ACTUALCOST
Type: Number

**CurrencyValue ->actualCostRateCurrency**
Required. Currency code for actual cost rate
*Table and Column:* PPA_WIP_VALUES.COST_CURRENCY_CODE
Type: String
CurrencyValue -> stdCostRate
  Required. The cost rate per unit.
  Table and Column: PPA_WIP_VALUES.STDCOST
  Type: Number

CurrencyValue -> stdCostRateCurrency
  Optional. The currency code for standard cost rate.
  Table and Column: PPA_WIP_VALUES.STDCOST_CURRENCY_CODE
  Type: String

CurrencyValue -> billRate
  Required. The billing rate per unit.
  Table and Column: PPA_WIP_VALUES.BILLRATE
  Type: Number

CurrencyValue -> billRateCurrency
  Required. The currency code for bill rate.
  Table and Column: PPA_WIP_VALUES.RATE_CURRENCY_CODE
  Type: String

CurrencyValue -> currencyType
  Required. Indicates the currency transaction in the values table. This picks up the HOME, BILLING & NATURAL transaction lines.
  Table and Column: PPA_WIP_VALUES.CURRENCY_TYPE
  Type: String

CurrencyValue -> totalCost
  Required. The ActualCost x quantity.
  Table and Column: PPA_WIP_VALUES.TOTALCOST
  Type: Number

CurrencyValue -> totalAmount
  Required. The (BillRate x quantity) + Factor + Burden + Overhead
  Table and Column: PPA_WIP_VALUES.TOTALAMOUNT
  Type: String

vendorCode
  Optional. The vendor code associated with the transaction.
  Table and Column: PPA_WIP_APINFO.VENDOR_CODE
  Type: String
notes
Optional. Additional information.

Table and Column: NOTES
Type: String

transactionStatus
Required. The status of the transaction.

Values:
■ 0. Normal
■ 1. Adjusted
■ 2. Reversed
■ 4. Under-adjust
■ 8. Under-bill

Table and Column: STATUS
Type: Number

CurrencyValue -> amountRemaining
Required. The Total Amount - Amount Billed.

Table and Column: PPA_WIP_VALUES.AMOUNTREMAINING
Type: Number

userLov1
Optional. Refers to PRTIMEENTRY_USERLOV1.

Table and Column: USER_LOV1
Type: String

userLov2
Optional. Refers to PRTIMEENTRY_USERLOV2.

Table and Column: USER_LOV2
Type: String

expenseType
Optional. Either CAPITAL_EXPENDITURE or DEPRECIATION.

Table and Column: EXPENSE_TYPE
Type: String
Transaction Import Schema Tag

The transaction import tag is part of the schema mapping for the Financial Transaction XOG object. It has the following attributes:

**externalID**
- Required. The external identifier. The value must be unique.
- **Table and Column:** EXTERNALID
- **Type:** String

**clientID**
- Optional. The valid company ID. If the client ID is empty, the value is taken from the project.
- **Table and Column:** COMPANY_CODE
- **Type:** String

**projectID**
- Required. The valid project ID.
- **Table and Column:** PROJECT_CODE
- **Type:** String

**taskId**
- Required. The valid internal task ID.
- **Table and Column:** TASKID
- **Type:** Number

**transactionDate**
- Required. The date of the transaction. It must be between the project start and end dates.
- **Table and Column:** TRANSDATE
- **Type:** Date

**resourceID**
- Optional. The resource ID associated with the transaction, when applicable. Otherwise, the resource ID from Cost Key Definition default.
- **Table and Column:** RESOURCE_CODE
- **Type:** String
roleID
Optional. The valid role identifier.
Table and Column: ROLE_CODE
Type: String

transactionType
Required. Defines the transaction type.
Values: L, M, X, and Q
Table and Column: TRANSTYPE
Type: String

chargeCode
Optional. A valid charge code.
Table and Column: CHARGE_CODE
Type: String

inputTypeCode
Optional. A valid input type.
Table and Column: INPUT_TYPE
Type: String

Chargeable
Optional. Indicates if the transaction is chargeable to the company.
Values:
- 0. Not chargeable
- 1. chargeable
Default: 0
Table and Column: CHARGEABLE
Type: Number

Units
The number of units (for all expense transactions). Not required for expense transactions, but required for all other transaction types.
Values: L, M, and Q (Quantity)
Table and Column: QUANTITY
Type: Number
actualCostRate
Optional. Cost rate per unit. It must be a valid number. If not specified, the cost is taken from the rate matrix.
*Table and Column:* PPA_WIP_VALUES.ACTUALCOST
*Type:* Number

actualCostRateCurrency
Optional. ISO currency code for actual cost rate.
*Table and Column:* PPA_WIP_VALUES.COST_CURRENCY_CODE
*Type:* String

BillRate
Optional. The billing rate per unit; must be a valid number; if not specified, rate will be picked up from the rate matrix.
*Table and Column:* RATE
*Type:* Number

billRateCurrency
Optional. ISO currency code for the bill rate.
*Table and Column:* RATE_CURRENCY
*Type:* String

notes
Optional. Additional information.
*Table and Column:* NOTES
*Type:* String

importStatus
Optional. The allowed value is N for New.
*Table and Column:* IMPORTSTATUS
*Type:* String

importDate
Optional. The date the transaction was imported. If not specified, the current server date is used.
*Table and Column:* IMPORTDATE
*Type:* Date
**General Ledger Account**

The general ledger (GL) account XOG object represents the chart of accounts used to process chargebacks. Use this XOG object to view inbound and outbound general ledger account attributes.

**Schema Name**

nikuxog_glaccount.xsd

**Read and Write XML Files**

The following XML files are included:

- pac_glaccount_read.xml. Use this file to export GL accounts from CA Clarity.
- pac_glaccount_write.xml. Use this file to import GL accounts that were previously exported from CA Clarity.

**Prerequisites**

The following conditions must be met before using this XOG:

- The entities referenced by the GL Accounts must exist prior to importing GL accounts.
- The Account Class must be a valid lookup value in CA Clarity.
- The Account Type must be a valid lookup value in CA Clarity.

**Business Rules and Processing**

The following business rules and processing apply to this XOG object:

- The GL chart of accounts is imported into CA Clarity from an external accounting system. These accounts are used by the Chargeback feature to capture charges and credits.
- The GL accounts schema is defined as part of the first step to GL integration.
- To enable GL transactions between systems, CA Clarity allows for inbound processing to define GL Accounts, Periods, and Entities. GL Accounts establishes the accounts to which transactions can be posted.

**Read Filters**

The following explicit read filters are used:

- **MAIN_ACCOUNT_ID.** This is used for filtering by mainAcctId, which is a part of the natural GL account code.
- **SUB_ACCOUNT_ID.** This is used for filtering by subAcctId, which is a part of the natural GL account code.
ACCOUNT_DESCRIPTION. This is used for filtering by description.

**Error Handling**

If a GL Account file is unsuccessful due to an error, the following fields are output:
- entity
- accountNumber
- externalId
- externalSource

**Schema Mappings**

The following schema mapping is provided for the outbound Financial Transaction tag name, GLAccount:
- **GLAccount** (see page 211)

**GLAccount Schema Tag**

This tag is part of the schema mapping for the General Ledger XOG object. The values in this table are unlike other lookup values. A text string must be provided, not a lookup code.

The GLAccount schema tag has the following attributes:

- **entity**
  
  Defines the entity name for the GL account code. This attribute is a unique primary key, browse field. Lookup to DPT_ENTITY_DEPT.

  **Table and Column:** Entity
  
  **Type:** Lookup

- **mainAcctId**
  
  Required. Part of the natural GL account code.

  **Table and Column:** MAIN_ACCOUNT_ID
  
  **Type:** String

- **subAcctId**
  
  Required. Part of the natural GL account code.

  **Table and Column:** SUB_ACCOUNT_ID
  
  **Type:** String
**Overhead**
Optional. Indicates if the GL account is an overhead account.

**Table and Column:** OVERHEAD  
**Type:** Boolean

**CapitalExpense**
Indicates if the GL account is a capital expense account.

**Table and Column:** CAPITAL_EXPENSE  
**Type:** Boolean

**noncashExpense**
Indicates if the GL account is a non-cash expense account.

**Table and Column:** NONCASH_EXPENSE  
**Type:** Boolean

**description**
Defines the description of the general ledger account.

**Table and Column:** Description  
**Type:** String

**Account Type**
Required. Defines the GL account type. This attribute determines whether the GL account is a Balance Sheet account or a P&L account. A default is set in the background. Lookup to PAC_CHG_GL_ACCOUNT_TYPE

**Table and Column:** ACCOUNT_TYPE  
**Type:** Number  
**Default:** Lookup

**accountClass**
Optional. This attribute determines whether the GL account is an asset or liability. A default is set in the background. Lookup to PAC_CHG_GL_ACCOUNT_CLASS.

**Default:** 0

**Table and Column:** ACCOUNT_CLASS  
**Type:** Lookup
Active
Indicates if the general ledger account is active.
Values:
- 0. Not Active
- 1. Active
Table and Column: Active
Type: Boolean
Default: 1

externalID
Defines the originating unique identifier.
Table and Column: External_ID
Type: String

externalSource
Defines the external source. The lookup value is the originating system ID.
Example: Oracle
Table and Column: External_Source_ID
Type: String in Schema, but stored as Number in CA Clarity.
General Ledger Allocation Rule

The GL Allocation Rule XOG object represents the debit and credit rules in chargebacks. Use the General Ledger Allocations Rule XOG object to import and export GL allocation rules.

Schema Name

nikuxog_glallocation.xsd

Read and Write XML Files

The following XML files are included:

- cbk_allocation_read.xml. Use this file to export GL allocations from CA Clarity.
- cbk_allocations_read.xml. Use this file to import GL allocations that were previously exported from CA Clarity.

Prerequisites

Before importing the GL allocation rules, make sure the entities referenced by these rules exist.

Business Rules and Processing

The insert or update of GL Allocation rules are based on the existence of the GL Allocation code in CA Clarity. The GL Allocation code is unique.

Read Filters

The following explicit read filters are used:

- ALLOCATION_CODE. A unique code that is used for filtering by the allocation code.
- STATUS. This is used for filtering by status (Active, Inactive, or On Hold)
- CBK_TYPE. This is used for filtering by chargeback type (Debit, Credit)
- CBK_SUB_TYPE. This is used for filtering by chargeback sub type (Standard, Investment or Overhead)

Schema Mapping

The following schema tag is included:

- GL Allocation Rule (see page 215)
GL Allocation Rule Schema Tag

The GL allocation rule tag is part of the schema mapping for the General Ledger Allocation Rule XOG object. It has the following attributes:

- **entityCode**
  - Optional. Defines the unique identifier of the entity tied to the GL allocation rule.
  - **Table and Column**: ENTITY_ID
  - **Type**: String

- **locationCode**
  - Optional. Defines the location unique identifier tied to the GL allocation rule.
  - **Table and Column**: LOCATION_ID
  - **Type**: String

- **departmentCode**
  - Optional. Defines the department unique identifier tied to the GL allocation rule.
  - **Table and Column**: DEPARTMENT_ID
  - **Type**: Boolean

- **resourceClassCode**
  - Optional. Defines the resource class unique identifier tied to the GL allocation rule.
  - **Table and Column**: RESOURCECLASS_ID
  - **Type**: Boolean

- **chargeCode**
  - Optional. Defines the charge code unique identifier tied to the GL allocation rule.
  - **Table and Column**: PRCHARGECODE_ID
  - **Type**: Boolean

- **investmentCode**
  - Optional. Defines the investment unique identifier tied to the GL allocation rule.
  - **Table and Column**: INVESTMENT_ID
  - **Type**: Number

- **utilityCode1**
  - Optional. The lookup to BROWSE_USR_VAL1_ALL.
  - **Table and Column**: UTILITY_CODE_1
  - **Type**: Lookup
utilityCode2
Optional. The lookup to PRTIMEENTRY_USERLOV2.
**Table and Column:** UTILITY_CODE_2
**Type:** Lookup

transactionClassCode
Optional. The lookup to FIN_TRANSCLASSES.
**Table and Column:** TRANSCLASS
**Type:** Lookup

typeCode
Optional. The lookup to LOOKUP_INPUT_TYPES.
**Table and Column:** PRTYPECODE_unique identifier
**Type:** Lookup

statusCode
Required. The lookup to STATUS_CODE.
**Values:** Open and Closed
**Table and Column:** PAC_CHG_STATUS
**Type:** Lookup

chargeRemToOverhead
Optional. Indicates if the rule charges the reminder to overhead.
**Table and Column:** CHG_REM_TO_OVERHEAD
**Type:** Boolean

cbkType
Required. Specifies the chargeback type.
**Values:** DEBIT and CREDIT
**Table and Column:** CHARGEBACK_TYPE
**Type:** String

cbkSubtype
Required. Specifies the chargeback subtype.
**Values:** STANDARD, INVESTMENT and OVERHEAD
**Table and Column:** CHARGEBACK_SUBTYPE
**Type:** String
Allocation Details

`glAccountMain, glAccountSub`
Required. Defines the main GL account. Lookup to `SCH_BROWSE_GL_ACCTS`.

**Table and Column:** GL_ACCOUNT_ID  
**Type:** Lookup

`department`
Required. Defines the unique identifier of the department to charge. Lookup to `SCH_BROWSE_DEPT`.

**Table and Column:** DEPARTMENT_ID  
**Type:** Lookup

`flatAmount`
Optional. This attribute is not used.

**Table and Column:** FLAT_AMOUNT  
**Type:** Numeric

`weightable`
Optional. This attribute is not used.

**Table and Column:** WEIGHTAGE  
**Type:** Numeric
General Ledger Period

Use the general ledger period XOG object to view inbound and outbound general ledger period attributes.

Schema Name

xog_glperiod.xsd

Read and Write XML Files

The following XML files are included:
- pac_glperiod_read.xml. Use this file to export GL periods from CA Clarity.
- pac_glperiod_write.xml. Use this file to import GL periods that were previously exported from CA Clarity.

Business Rules and Processing

GL Periods are only defined for inbound (write) processing to CA Clarity. This schema is defined as part of the first step to GL integration.

Read Filters

None

Error Handling

If a GL Period file is unsuccessful due to an error, the following fields are output:
- Period Name
- Period Type
- Period Number
- Description
- Quarter
- Year
- Start Date

Schema Mappings

Schema mappings are described for the following outbound General Ledger Period tag name:
- Glperiod (see page 219)
Glperiod Schema Tag

This tag is part of the schema mapping for the General Ledger Period XOG object. It has the following attributes:

**entity**
- Required. The unique primary key. A browse field associating the period to an entity.
  
  **Table and Column:** Entity
  
  **Type:** String

**Period**
- Required. Defines the unique primary key. The fiscal period (i.e., date) posted for the selected entity.
  
  **Table and Column:** Period
  
  **Type:** Date

**currentPeriod**
- Required. Defines the status of the resource.
  
  **Values:**
  - 1. True
  - 0. False
  
  **Default:** 1
  
  **Table and Column:** Currentperiod
  
  **Type:** Boolean

**externalId**
- Required. The originating unique ID.
  
  **Table and Column:** External_ID
  
  **Type:** String

**externalSource**
- Required. A lookup value is the originating system ID (for example, Oracle).
  
  **Table and Column:** External_Source_ID
  
  **Type:** String in schema, but stored as Number in the database.
GL transactions represents an entry in the General Ledger. It includes information such as the accounts credited or debited and other financial transaction information. Use the general ledger transaction XOG object to view outbound general ledger attributes.

Schema Names

nikuxog_transaction.xsd

Read and Write XML Files

The following XML files are included:

- pac_gltransactions_read.xml. Use this file to export GL transactions from CA Clarity.
- pac_gltransactions_write.xml. Use this file to import GL transactions that were previously exported from CA Clarity.

Prerequisites

The GL transaction must belong to an invoice.

Business Rules and Processing

The GL Transactions schema is defined for outbound (read) GL processing. The GL Transaction object is used to export the data from the CBK_GL_TXNS and CBK_GL_TXN_VALUES tables.

Read Filters

The following explicit read filters are used:

- transactionSource. This is used to filter GL transactions by the transaction source (W for WIP, A for Adjusted, or R for Reversed).
- entity. This is used to filter by GL transactions by entity.
- periodStart. This is used to filter GL transactions by the fiscal period start date, a date filter.
- periodEnd. This is used to filter GL transactions by the fiscal period end date, a date filter.
- investment_id. This filter is used to filter GL Transactions by the investment.

XOG allows for outbound processing of GL Transactions based upon the value within the glposted field.
When querying the database, by default the query returns all GL transactions where glposted is not equal to 'Y', (that is, transactions are awaiting posting). Once selected and invoices processes, the glposted field is updated to "Y" to indicate they have been sent to the GL.

**Error Handling**

**Read Transactions**

Error handling for read transactions from CA Clarity databases are due to an invalid formats or database unavailability. The adaptor or middleware must handle transaction-level error handling when mapping and transporting into the accounting system. If one transaction is found to be in error, the entire file is not committed. The file must be fixed and resubmitted to keep the balance of debits and credits.

XOG does not have control of processing once an output file is successfully created. If you find an error in the output, you will need to rollback the entire batch to keep debits and credits intact.

If a single record within the batch is found to be in error, the entire batch is rejected. Then:

- The external system (adaptor or middleware) must call the Update Transactions schema and provide the error information element tag and the key fields of the error records.
- XOG processes the input file and copies all the GL transaction records from the GLCONTROL table into the GLEXCEPTION table.
- XOG deletes the records from the GL Control table so they exist in the GLEXception table.
- XOG resets the GLPOSTED field for the transactions from the batch in the PPA_WIP and PPA_BILLING table to 'N' from 'P' (depending if the transaction source is B or W).

You must fix the error batches via CA Clarity, re-post to GLControl, and rerun the XOG to extract the GL transaction records.

**Update Transactions**

If the entire file cannot be committed, it must be fixed and resubmitted. This is important as all debits and credits must be kept in sync across applications. If an error is found, it is written to the error log. The following fields help to identify the transaction in error:

- transactionNumber
- transactionSource
- sequenceNumber
Schema Mappings

Schema mappings are described for the following outbound General Ledger Transaction
tag name:

- GLtransaction (see page 222)

GLtransaction Schema Tag

This tag is part of the schema mapping for the General Ledger Transaction XOG object. It
has the following attributes:

**entity**

Optional. The name of the entity for the GL transaction.

**Table and Column:** CBK_GL_TXNS.ENTITY_ID

**Type:** String

**accountCode**

Required. The GL account code.

**Table and Column:** CBK_GL_TXNS.GL_ACCOUNT_ID

**Type:** String

**amount**

Optional. The amount of the transaction.

**Table and Column:** CBK_GL_TXN_VALUES.AMOUNT

**Type:** Float

**currency**

Optional. The currency code of the transaction amount.

**Table and Column:** CBK_GL_TXN_VALUES.CURRENCY_TYPE,  
CBK_GL_TXN_VALUES.CURRENCY_CODE

**Type:** String

**transactionNumber**

Required. A unique primary key. The transaction number from WIP or PPA-billings.

**Table and Column:** CBK_GL_TXN_VALUES.TRANSACTION_ID

**Type:** Positive Integer
**transactionSource**

Required. A unique primary key. This allows you to define the GL distribution of a transaction based on the module where it originated.

**Values:**
- W. From WIP.
- A. From billing.
- D. From credit.

**Table and Column:** TRANSACTION_SOURCE

**Type:** String

**period**

Required. The gl period for the transaction.

**Table and Column:** PPA_WIP.GLPERIOD

**Type:** String

**InvoiceDate**

Required. The date of the invoice to which the GL transaction belongs. It must be between the project start and end dates.

**Table and Column:** CBK_INVOICE.INVOICE_DATE

**Type:** Date

**department**

Required. The department id of the transaction

**Table and Column:** CBK_GL_TXNS(DEPARTMENT_ID

**Type:** String

**transactionDate**

Required. The date of the transaction.

**Table and Column:** CBK_GL_TXNS.TRANSACTION_DATE

**Type:** Date

**investment**

Required. The investment on which the transaction is posted.

**Table and Column:** CBK_GL_TXNS.TRN_INV_ID

**Type:** String
Idea

The Idea XOG object extends the common investment object as Ideas are a type of investment. Use this XOG object to view inbound (write) and outbound (read) idea processing.

Schema Names

- **nikuxog_idea.xsd.** Use to view inbound idea attributes. It determines which XML elements and attributes are required to import idea information from an external system into CA Clarity.
- **nikuxog_read.xsd.** Use to view outbound idea attributes. It determines the XML format required to export idea information from CA Clarity to another system.

Read and Write XML Files

The following XML files are included:

- **ideas_read.xml.** Use this file to export idea object instances from CA Clarity.
- **ideas_write.xml.** Use this file to import idea object instances that were previously exported from CA Clarity.

Prerequisites

The manager username should exist.

Business Rules and Processing

Prior to importing ideas, the following items must be correctly set up:

**ID**

The unique identifier for the idea. If the ID does not exist, the XML schema creates a new idea record unless auto-numbering is enabled (then an error is generated and posted to the Success and Error files). If the XOG input includes an idea ID, and an idea with the same ID is found, that record is updated.

**Summary**

The summary of the idea.

**Finish date**

The completion date for the idea. If a finish date exists, there must also be a start date. The finish date must also be greater than the start date.

**Break-even date**

This is the date when the idea is to break-even. If a break-even date exists, it must be greater than or equal to the start date.
Read Filters

The following explicit read filters are used with this XOG:

- objectId
- managerUserName
- lastUpdatedDate

Error Handling

Manager username must be set to NULL if it does not exist in CA Clarity.

Schema Mappings

Idea is composed of the following elements, which are inherited from the investment object, and idea-specific mapping found in the Idea Schema tag:

- Allocations
- scenarioDependencies
- InvestmentAssociations
- InvestmentBaselines
- InvestmentResources
- InvestmentTasks
- General
- OBSAssocs
- CustomInformation

Ideas are different from other investments because they do not include child investments. You can only associate ideas with a simple budget, and they do not have full financial planning capabilities like other investment types.

Idea Schema Tag

This tag is part of the schema mapping for the Idea XOG object. It has the following attributes:

**Note:** The import validation rule applies only if it is different from the object attribute-level validation rule
Idea

**ideaspriority**
Optional. Defines the priority for the idea.

*Table and Column:* INV_IDEAS.PRIORITY
*Values:* High, Medium, Low

**estimateType**
Optional.

*Table and Column:* INV_IDEAS.EST_TYPE
*Type:* Number
*Values:* Historical, Analytical, High Level, and Commitment

**estimatedCost**
Optional.

*Table and Column:* INV_IDEAS.EST_COST
*Type:* nonNegativeDouble

**estimatedBenefit**
Optional.

*Table and Column:* INV_IDEAS.EST_BENEFIT
*Type:* nonNegativeDouble

**benefitDescription**
Optional.

*Table and Column:* INV_IDEAS.BENEFIT_DESC
*Type:* String

**generalNotes**
Optional.

*Table and Column:* INV_IDEAS.GENERAL_NOTES
*Type:* String

**businessUnit**
Optional.

*Table and Column:* INV_IDEAS.BUS_UNIT
*Type:* String
breakevenDate
Optional.
Table and Column: INV_IDEAS.BREAKEVEN_DATE
Type: investmentDateTimeType

impact
Optional.
Table and Column: INV_IDEAS.IMPACT
Type: String

risks
Optional.
Table and Column: INV_IDEAS.RISKS
Type: String

dependencies
Optional.
Table and Column: INV_IDEAS.DEPENDENCIES
Type: String

estimatedStartDate
Optional.
Table and Column: INV_IDEAS.EST_START_DATE
Type: investmentDateTimeType

estimatedFinishDate
Optional.
Table and Column: INV_IDEAS.EST_FINISH_DATE
Type: investmentDateTimeType

conversionDate
Optional.
Table and Column: INV_IDEAS.CONVERSION_DATE
Type: investmentDateTimeType

createdDate
Optional.
Table and Column: INV_IDEAS.PMA_IDEA.CREATED_DATE
createdBy
Optional.
  **Table and Column:** INV_IDEAS.PMA_IDEA.CREATED_BY

lastUpdatedDate
Optional.
  **Table and Column:** INV_IDEAS.PMA_IDEA.LAST_UPDATED_DATE

lastUpdatedBy
Optional.
  **Table and Column:** INV_IDEAS.PMA_IDEA.LAST_UPDATED_BY

ideasPriority
Optional.
  **Table and Column:** INV_IDEAS.PRIORITY
  **Type:** Number
  **Import Validation Rule:** The priority range used in ideas differs from the range used for investments.

targetManagerUserName
Optional.
  **Table and Column:** INV_IDEAS.TARGET_MANAGER_ID
  **Type:** String
  **Import Validation Rule:** The name of the manager who is targeted to manage the investment after conversion from an idea.

**Investments Schema Tag**

This tag is part of the schema mapping for the Idea XOG object. The INV_INVESTMENTS table contains the shared investment attributes. This tag has the following attributes:

status
Optional. Defines the status of the investment.
  **Values:** Open, Unapproved, Rejected, Approved, Incomplete, Submitted for Approval, and Converted
  **Default:** Open
  **Table and Column:** INV_INVESTMENTS.STATUS
  **Type:** Number
<table>
<thead>
<tr>
<th>Name</th>
<th>Required/Optional</th>
<th>Table and Column</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Required</td>
<td>INV_INVESTMENTS.NAME</td>
<td>String</td>
</tr>
<tr>
<td>objectID</td>
<td>Optional</td>
<td>INV_INVESTMENTS.CODE</td>
<td>String</td>
</tr>
<tr>
<td>description</td>
<td>Optional</td>
<td>INV_INVESTMENTS.DESCRIPTION</td>
<td>String</td>
</tr>
<tr>
<td>priority</td>
<td>Optional</td>
<td>INV_INVESTMENTS.PRIORITY</td>
<td>Number</td>
</tr>
<tr>
<td>managerUserName</td>
<td>Optional</td>
<td>INV_INVESTMENTS.MANAGER_ID</td>
<td>String</td>
</tr>
<tr>
<td>approvedById</td>
<td>Optional</td>
<td>INV_INVESTMENTS.APROVEDBY_ID</td>
<td>String</td>
</tr>
<tr>
<td>chargeCodeExtID</td>
<td>Optional</td>
<td>INV_INVESTMENTS.CHARGECODEID</td>
<td>String</td>
</tr>
<tr>
<td>approvedTime</td>
<td>Optional</td>
<td>INV_INVESTMENTS.APROVEDTIME</td>
<td>investmentDateTimeType</td>
</tr>
</tbody>
</table>
processCode
Optional.
**Table and Column:** INV_INVESTMENTS.PROCESS_CODE
**Type:** investmentCodeType

stageCode
Optional.
**Table and Column:** INV_INVESTMENTS.STAGE_CODE
**Type:** investmentCodeType

goalCode
Optional.
**Table and Column:** INV_INVESTMENTS.GOAL_CODE
**Type:** investmentCodeType

alignment
Optional.
**Table and Column:** INV_INVESTMENTS.ALIGNMENT
**Type:** iNumber

risk
Optional.
**Table and Column:** INV_INVESTMENTS.RISK
**Type:** Number

statusIndicator
Optional.
**Table and Column:** INV_INVESTMENTS.STATUS_INDICATOR
**Type:** Number

statusComment
Optional.
**Table and Column:** INV_INVESTMENTS.STATUS_COMMENT
**Type:** String
progress
Optional.
**Default:** 0 - Not Started
**Table and Column:** INV_INVESTMENTS.SPROGRESS
**Type:** Number

currencyISOcode
Optional.
*Note: This attribute replaces currencyCode*
**Table and Column:** INV_INVESTMENTS.CURRENCY_CODE
**Type:** String

**Budget Schema Tag**

This tag is part of the schema mapping for the Ideas XOG object. The FIN_FINANCIALS table contains the shared budget attributes.

The Budget schema tag has the following attributes:

**plannedCostTotal**
Optional.
**Table and Column:** FIN_FINANCIALS.PLANNED_CST_TOTAL
**Type:** nonNegativeDouble

**plannedCostStart**
Optional.
**Table and Column:** FIN_FINANCIALS.PLANNED_CST_START
**Type:** investmentDateTimeType

**plannedCostFinish**
Optional.
**Table and Column:** FIN_FINANCIALS.PLANNED_CST_FINISH
**Type:** investmentDateTimeType

**plannedBenTotal**
Optional.
**Table and Column:** FIN_FINANCIALS.PLANNED_BEN_TOTAL
**Type:** nonNegativeDouble
plannedBenStart
Optional.

Table and Column: FIN_FINANCIALS.PLANNED_BEN_START
Type: investmentDateTimeType

plannedBenFinish
Optional.

Table and Column: FIN_FINANCIALS.PLANNED_BEN_FINISH
Type: investmentDateTimeType

budgetCostTotal
Optional.

Table and Column: FIN_FINANCIALS.BUDGET_CST_TOTAL
Type: nonNegativeDouble

budgetCostStart
Optional.

Table and Column: FIN_FINANCIALS.BUDGET_CST_START
Type: investmentDateTimeType

budgetCostFinish
Optional.

Table and Column: FIN_FINANCIALS.BUDGET_CST_FINISH
Type: investmentDateTimeType

budgetCostOnHold
Optional.

Table and Column: FIN_FINANCIALS.BUDGET_CST_ONHOLD
Type: investmentDateTimeType

budgetCostResumed
Optional.

Table and Column: FIN_FINANCIALS.BUDGET_CST_RESUMED
Type: investmentDateTimeType

budgetRevTotal
Optional.

Table and Column: FIN_FINANCIALS.BUDGET_REV_TOTAL
Type: nonNegativeDouble
**budgetRevStart**

Optional.

*Table and Column:* FIN_FINANCIALS.BUDGET_REV_START

*Type:* investmentDateTimeType

**budgetRevFinish**

Optional.

*Table and Column:* FIN_FINANCIALS.BUDGET_REV_FINISH

*Type:* investmentDateTimeType

**budgetRevOnHold**

Optional.

*Table and Column:* FIN_FINANCIALS.BUDGET_REV_ONHOLD

*Type:* investmentDateTimeType

**budgetRevResumed**

Optional.

*Table and Column:* FIN_FINANCIALS.BUDGET_REV_RESUMED

*Type:* investmentDateTimeType

**forecastCostTotal**

Optional.

*Table and Column:* FIN_FINANCIALS.FORECAST_CST_TOTAL

*Type:* nonNegativeDouble

**forecastCostStart**

Optional.

*Table and Column:* FIN_FINANCIALS.FORECAST_CST_START

*Type:* investmentDateTimeType

**forecastCostFinish**

Optional.

*Table and Column:* FIN_FINANCIALS.FORECAST_CST_FINISH

*Type:* investmentDateTimeType

**forecastCostOnHold**

Optional.

*Table and Column:* FIN_FINANCIALS.FORECAST_CST_ONHOLD

*Type:* investmentDateTimeType
**forecastCostResumed**
Optional.
- **Table and Column:** FIN_FINANCIALS.FORECAST_CST_RESUMED
- **Type:** investmentDateTimeType

**forecastRevTotal**
Optional.
- **Table and Column:** FIN_FINANCIALS.FORECAST_REV_TOTAL
- **Type:** nonNegativeDouble

**forecastRevStart**
Optional.
- **Table and Column:** FIN_FINANCIALS.FORECAST_REV_START
- **Type:** investmentDateTimeType

**forecastRevFinish**
Optional.
- **Table and Column:** FIN_FINANCIALS.FORECAST_REV_FINISH
- **Type:** investmentDateTimeType

**forecastRevOnHold**
Optional.
- **Table and Column:** FIN_FINANCIALS.FORECAST_REV_ONHOLD
- **Type:** investmentDateTimeType

**forecastRevResumed**
Optional.
- **Table and Column:** FIN_FINANCIALS.FORECAST_REV_RESUMED
- **Type:** investmentDateTimeType

**plannedNPV**
Optional.
- **Table and Column:** FIN_FINANCIALS.PLANNED_NPV
- **Type:** Number

**plannedROI**
Optional.
- **Table and Column:** FIN_FINANCIALS.PLANNED_ROI
- **Type:** Number
plannedBreakEven

Optional.

*Table and Column:* FIN_FINANCIALS.PLANNED_BREAKEVEN

*Type:* investmentDateTimeType

budgetNPV

Optional.

*Table and Column:* FIN_FINANCIALS.BUDGET_NPV

*Type:* Number

budgetROI

*Table and Column:* FIN_FINANCIALS.BUDGET_ROI

*Type:* Number

budgetBreakEven

Optional.

*Table and Column:* FIN_FINANCIALS.BUDGET_BREAKEVEN

*Type:* investmentDateTimeType

setBudgetValuesEqualToPlannedValues

Optional.

*Table and Column:*

*Type:* Boolean

calculateFinancialMetrics

Optional.

*Table and Column:*

*Type:* Boolean

initialInvestment

This column refers to the initial, lump sum cost associated with an investment

*Table and Column:* FIN_FINANCIALS.INITIAL_INVESTMENT

*Type:* Number

plannedIRR

Internal Rate of Return on investment as calculated from the planned costs and planned benefits. This is used only in a XOG read result.

*Table and Column:* FIN_FINANCIALS.PLANNED_IRR

*Type:* Number
plannedMIRR

Modified internal rate of return on an investment as calculated from the planned costs and planned benefits. This is used only in a XOG read result.

**Table and Column:** FIN_FINANCIALS.PLANNED_MIRR

**Type:** Number

plannedPaybackPeriod

Payback period of investment as calculated from the planned costs and planned benefits. This is used only in a XOG read result.

**Table and Column:** FIN_FINANCIALS.PLANNED_PAYBACK_PERIOD

**Type:** Number

budgetIRR

Internal rate of return on an investment as calculated from the budgeted costs and budgeted benefits. This is used only in a XOG read result.

**Table and Column:** FIN_FINANCIALS.BUDGET_IRR

**Type:** Number

budgetMIRR

Modified internal rate of return on an investment as calculated from the budgeted costs and budgeted benefits. This is used only in a XOG read result.

**Table and Column:** FIN_FINANCIALS.BUDGET_MIRR

**Type:** Number

budgetPaybackPeriod

Payback Period of investment as calculated from the budgeted costs and budgeted benefits. This is used only in XOG read result.

**Table and Column:** FIN_FINANCIALS.BUDGET_PAYBACK_PERIOD

**Type:** Number

useSystemDefinedTotalCostOfCapital

This column indicates whether an investment uses the system-defined reinvestment rate % for financial metrics calculations.

**Table and Column:** FIN_FINANCIALS.IS_SYS_VAL_FOR_TCC

**Type:** Boolean

investmentDefinedTotalCostOfCapital

Investment-specific total cost of capital percentage used for financial metrics calculations based on the value of IS_SYS_VAL_FOR_TCC.

**Table and Column:** FIN_FINANCIALS.TOTAL_COST_OF_CAPITAL

**Type:** Number
**useSystemDefinedReinvestmentRate**

Indicates whether an investment uses the system-defined total cost of capital percentage for financial metrics calculations.

**Table and Column**: FIN_FINANCIALS.IS_SYS_VAL_FOR_RR

**Type**: Boolean

**investmentDefinedReinvestmentRate**

Investment-specific reinvestment rate percentage used for financial metrics calculations based on the value of IS_SYS_VAL_FOR_RR

**Table and Column**: FIN_FINANCIALS.REINVESTMENT_RATE

**Type**: String
Inbound Transaction

Use the inbound transaction XOG object to view inbound and outbound inbound transactions. Inbound transactions are the cost and/or revenue posted for tasks or investments.

Schema Name

nikuxog_inboundTransaction.xsd

Read and Write XML Files

The following XML files are included:
- imp_transactions_read.xml. Use this file to export inbound incident attributes from CA Clarity.
- imp_transactions_read.xml. Use this file to import inbound incident attributes that were previously exported from CA Clarity.

Prerequisites

None

Read Filters

The following explicit read filters are used:

- **projectID**
  - Defines the name of the investment.
- **clientID**
  - Defines the company unique identifier.
- **transactionSource**
  - Defines the transaction source.
- **transactionDate**
  - Defines the date of the transaction.

Error Handling

The following error can be thrown:
- **Wip Transaction Object read failed.**

Schema Mapping

Mappings for the following schema tag name is provided:
inboundTransactionType (see page 239)

**inboundTransactionType Schema Tag**

The inboundTransactionType tag is part of the schema mapping for the inbound transaction XOG object. This schema tag has the following attributes:

- **groupId**
  - Optional. The group unique identifier for the voucher entry.
  - Table and Column: PPA_TRANSCONTROLAPINFO.ID
  - Type: Double

- **voucherNumber**
  - Optional. The voucher number for the voucher entry.
  - Table - Field Name: PPA_TRANSCONTROLAPINFO.VOUCHERNO
  - Type: String

- **poNumber**
  - Optional. The PO number for the voucher entry.
  - Table and Column: PPA_TRANSCONTROLAPINFO.PONO
  - Type: String

- **vendorCode**
  - Optional. The vendor code for the voucher entry.
  - Table and Column: PPA_TRANSCONTROLAPINFO.VENDOR_CODE
  - Type: String

- **incurredBy**
  - Optional. The group Id for the voucher entry.
  - Table and Column: PPA_TRANSCONTROLAPINFO.INCURRED_BY
  - Type: String

- **externalId**
  - Optional. The external id for the voucher entry.
  - Table and Column: IMP_TRANSACTIONIMPORT.EXTERNAL_ID
  - Type: String
actualCostRate
  Optional. The cost rate for the voucher entry.
  **Table and Column:** IMP_TRANSACTIONIMPORT.COST
  **Type:** Double

actualCostRateCurrency
  Optional. The actual cost rate currency for the voucher entry.
  **Table and Column:** IMP_TRANSACTIONIMPORT.COST_CURRENCY
  **Type:** String

billRate
  Optional. The bill rate for the voucher entry.
  **Table and Column:** IMP_TRANSACTIONIMPORT RATE
  **Type:** Double

billRateCurrency
  Optional. The billing rate currency for the voucher entry.
  **Table and Column:** IMP_TRANSACTIONIMPORT RATE_CURRENCY
  **Type:** String

importStatus
  Optional. The import status for the voucher entry. The only value allowed is 'N'
  which means New.
  **Table and Column:** IMP_TRANSACTIONIMPORT.IMPORTSTATUS
  **Type:** String
Incident

Use the incident XOG object to view inbound and outbound incident instances.

Schema Name

nikuxog_incident.xsd

Read and Write XML Files

The following XML files are included:

- incident_read.xml. Use this file to export incident object instances from CA Clarity.
- incident_write.xml. Use this file to import incident object instances that were previously exported from CA Clarity.

Prerequisites

The referred Investment must exist in the system.

Read Filters

None

Error Handling

The following errors can be thrown:

- Required fields: Ensures all required fields have values.
- Incident type is not valid.
- Status is not valid.
- Priority is not valid.
- Urgency is not valid.
- Impact is not valid.
- Category is not valid.
- Assigned To Code is not valid.
- Reported By Code is not valid.
- Assigned Project Manager Code is not valid.
- External Source is not valid.
- Investment Object {0} is not valid.
- Resource for Effort is not valid.
- Failed to export incident.
Failed to export category.
Failed to import incident.
Failed to import category.
Estimated Time to Complete cannot be negative.
Investment ID is not valid.
Required attribute categoryCode is missing.

Schema Mapping

The following incident schema tag names are provided to XOG incidents:

- **incidents** (see page 242)
- **Description** (see page 243)

**incidents Schema Tag**

The incidents tag is part of the schema mapping for the incident XOG object. This is a placeholder tag for multiple incidents.

**incident**

The actual incident object. Incident has the following attributes:

**incident_code**

Required. Defines the unique incident code.

*Table and Column:* incidents.departcode

*Type:* String

**short_description**

Required. Defines the incident name.

*Table and Column:* incidents.shortdesc

*Type:* String

**dept_identifier**

Defines the general ledger segment value mapped to this incident.

*Table and Column:* incidents.departidentifier

*Type:* String
default_reviewer
Defines the default reviewer for the incident.
Table and Column: incidents.default_reviewer
Type: String

alt_default_reviewer
defines the alternate reviewer for the incident.
Table and Column: incidents.alt_default_reviewer
Type: String

parent_incident_code
Defines the code for parent incident.
Table and Column: parent_incident_id
Type: String

department_manager_code
The incident manager resource code.
Table and Column: incidents.incident_manager_id
Type: String

brm_code
Defines the business relationship manager.
Table and Column: brm_id
Type: String

entity
Required. The identify for the entity to which the incident belongs.
Table and Column: incidents.entity_id
Type: String

delegate_invoice_approval
Indicates if the incident delegates invoice approval.
Table and Column: incidents.delegate_inv_appr
Type: Integer

incidents Schema Tag
The description tag is part of the schema mapping for the incident XOG object. This is a placeholder tag for multiple incidents, and has the following attributes:
description

Optional. Defines the description of the incident.

**Table and Column:** IMM_INCIDENTS.DESCRIPTION

**Type:** String
Investment

The investment XOG object is used by multiple objects all of which share a common foundation. The nikuxog_inv_common.xsd is a common schema shared among all investments. The schema definition contained in this file along with the extended schema definition in a specialized investment file together make up the schema for a particular investment type.

The investment object is an abstract object that you can only create as one of the following objects which extend the base investment object:

- Project
- Classes

Schema Name

nikuxog_inv_common.xsd

Read and Write XML Files

A generic read and write xml files is not included for investments. You can read or write investments in their specific investment type form, such as application, asset, or product. The following files contains part of the business logic used to read investments:

- inv_common_read.xbl. Use this file to export investments from CA Clarity.
- inv_common_write.xbl. Use this file to import investments that were previously exported from CA Clarity.

Prerequisites

Any referenced resources (for example, managerName) should exist in CA Clarity.

Business Rules and Processing

Objects which are based upon the Investment object are defined for inbound (write) and outbound (read) processing. The abstract Investment object itself cannot be written to or read from CA Clarity.

Read Filters

The following explicit read filters are used:

- objectID. The investment object’s unique ID.
- managerUserName. The name of the investment manager.
- lastUpdatedDate. The date when the investment was last modified.

Error Handling
The following errors can be thrown:

- Exports are truncated if the number of Application objects retrieved is larger than the system default: 5000. This setting can be overridden by setting the argument `args_maxrecords` with a new max.
- Charge code does not exist in the System.
- Goal code does not exist in the System.
- Category code does not exist in the System.
- Process code does not exist in the System
- Stage code does not exist in the System.
- Approver does not exist in the System.
- Currency must be active; on a multicurrency system it must have an exchange rate configured against the base currency.
- Entity Code 'XXX' does not exist in the System at this time. Please re-run the XOG if it exists now.
- Invalid Chargeback Type with name 'XXX'. This Chargeback Type could not be found in the System.
- Invalid Bill Expense Type with name 'XXX'. This Bill Expense Type could not be found in the System.
- XOG user does not have Approval right for objects of type 'XXX'; status of 'YYY' remains unchanged.
- Application with code 'XXX' does not exist in the System, and hence the Service is not updated with Application Code.
- Target Manager User Name 'XXX' passed in does not exist in the system.
- Investment has financial plans hence entityCode value "XXX" cannot be changed to YYY"
- Investment has posted transactions hence entityCode value "XXX" cannot be changed to YYY".
- OBS is Lowest Unit for 'XXX'. This unit path is not the lowest unit in its branch
- OBS unit (XXX) is invalid.
- OBS unit path (XXX) is invalid. So, the OBS Association was not made.
- OBS id (XXX) is invalid. So, the OBS Association was not made.
- OBS unit path is invalid. So the OBS security was not setup.
- OBS unique name is invalid. So the OBS security was not setup.
- Group is invalid. So the Group security was not setup.
- User is invalid. So the User security was not setup.
- {$departcode} and {$locationid} are not associated
- {$locationid} cannot be found
- {$departcode} cannot be found
- Rate Matrix with name XXX referenced in field YYY does not exist in the System at this time. Please re-run the XOG if it exists now.
- Invalid Exchange Rate Type with name XXX found for YYY. Default value Average will be used instead.
- The allocation status: 'XXX' is invalid.
- The chargeback type: 'XXX' is invalid. Should be DEBIT or CREDIT.
- The chargeback subtype: 'XXX' is invalid. Should be STANDARD or INVESTMENT or OVERHEAD.
- The chgRemtoOverhead has to be 0 or 1.
- The allocation code is required.
- Cannot determine allocation type from {$cbk_type} and {$cbk_subtype}.
- Resource Class -- XXX is invalid.
- Charge Code -- XXX is invalid.
- Investment -- XXX is invalid.
- Investment -- XXX contains Allocation -- YYY that is already in use by another investment.
- Investment -- XXX contains Allocation -- YYY that is already in use by global allocation.
- User Value 1 -- XXX is invalid.
- User Value 2 -- XXX is invalid.
- Transaction Class -- XXX is invalid.
- Input Type -- XXX is invalid.
- Gl Account -- {$acctMain} - {$acctSub} is invalid.
- Department -- XXX is invalid.
- The Scenario Dependency specified - XXX- would cause a circular reference; not added.
- The Scenario Dependency specified - XXX - is not active or does not exist.
- Missing Edit permission on XXX object; Scenario Dependencies not added.
- Specified parent investment XXX does not exist in the System at this time. Please re-run the XOG if it exists now.
- Unable to add XXX as a parent investment, due to circular dependencies.
- Unable to add XXX as a child investment, due to circular dependencies.
Specified child investment XXX does not exist in the System at this time. Please re-run the XOG if it exists now.

Investment Object operation failed while processing Investment Association.

No match could be found for Assignment teamId = XXX.

When specifying assignments for multiple roles, a teamId is required. role = XXX.

Could not determine the teamUid for assignment resource = XXX.

Team resource does not exist XXX.

Assignment role XXX does not exist.

-- Task XXX has no taskID; Estimate Rules are not imported.

Constraint date attribute cannot be null when delete flag is null. Could not add constraint with type = XXX!

Constraint type attribute is not a correct value. Could not add constraint with type = XXX.

Constraint type attribute is required! Could not add constraint.

When inserting multiple roles for the same role resourceId you must supply a teamId.

Investment Management Object XXX failed.

Schema Mapping

The Investment XOG object is composed of the following:

- Investment
- Allocations
- Details
- scenarioDependencies
- InvestmentAssociations
- InvestmentBaselines
- UsageCurve and CostCurve
- InvestmentResources
- InvestmentTasks
- General
- OBSAssocs
- Custom Information
Investment Schema Tag

This tag is part of the schema mapping for the investment XOG object. The actual tag is Asset, Idea, Project, or any other investment object. The abstract investment object includes attributes related to general properties of the investment, budgeting for the investment, and financial charges information.

An investment is composed of the following sub elements:

- Allocations
- scenarioDependencies
- InvestmentAssociations
- InvestmentBaselines
- InvestmentResources
- InvestmentTasks
- General
- OBSAssocs
- CustomInformation

For the Idea object, the Allocations element is not supported and InvestmentAssociations may only include parents.

The following tables are references in the Investment schema tag:

- INV_INVESTMENTS table contains the shared investment attributes.
- FIN_FINANCIALS table contains the shared budget attributes.
- PAC_MNT_PROJECTS table contains the shared financial attributes (not applicable for Idea)

The Investment schema tag has the following attributes:

**ID**

Required. Specifies the unique ID for the investment.

**Table and Column:** PROJCLASS.ID  
**Type:** Number

**investmentClass**

Required. Specifies the investment class for the investment.

**Table and Column:** PROJCLASS.PROJCLASS  
**Type:** String
status
Optional. Defines the status of the investment.
Values: Unapproved, Approved, Rejected, and Cancelled
An idea has three additional statuses: Submitted For Approval, Incomplete, and Converted
Table and Column: INV_INVESTMENTS.STATUS
Type: Number

name
Required. The name of the investment.
Table and Column: INV_INVESTMENTS.NAME
Type: String

objectID
Optional. Autonumbering is required if it is not present.
Table and Column: INV_INVESTMENTS.CODE
Type: String

description
A text description of the investment.
Table and Column: INV_INVESTMENTS.DESCRIPTION
Type: String

shortdesc
A short description of the investment.
Table and Column: PROJCLASS.SHORTDESC
Type: String

priority
Optional. Defines the priority.
Values: 0-36, where:
- 0. Highest priority
- 36. Lowest priority
Table and Column: INV_INVESTMENTS.PRIORITY
Type: Number
managerUserName
Optional. The name of the manager for the investment.
**Table and Column:** INV_INVESTMENTS.MANAGER_ID
**Type:** String

approvedById
Optional. The name of the CA Clarity user who approved the investment.
**Table and Column:** INV_INVESTMENTS.APPROVEDBY_ID
**Type:** String

chargeCodeExtID
Optional.
**Table and Column:** INV_INVESTMENTS.CHARGECODEID
**Type:** String

createdDate
Optional. Specifies the date the investment was created.
**Table and Column:** PROJCLASS.CREATED_DATE
**Type:** Date

createdBy
Optional. Specifies who created the investment.
**Table and Column:** PROJCLASS.CREATED_BY
**Type:** String

updatedBy
Optional. Specifies who last updated the investment.
**Table and Column:** PROJCLASS.UPDATED_BY
**Type:** String

lastUpdatedBy
Optional. The name of the user who last modified the investment.
**Table and Column:** INV_INVESTMENTS.LAST_UPDATED_BY
**Type:** String

lastUpdatedDate
Optional. The date and time when the investment was last modified.
**Table and Column:** INV_INVESTMENTS.LAST_UPDATE_DATE
**Type:** investmentDateTimeType
approvedTime
Optional. The date and time when the investment was approved.

Table and Column: INV_INVESTMENTS.APPROVEDTIME
Type: investmentDateTimeType

processCode
Optional. The process associated with the investment.

Table and Column: INV_INVESTMENTS.PROCESS_CODE
Type: investmentCodeType

stageCode
Optional. The stage within the investment process.

Table and Column: INV_INVESTMENTS.STAGE_CODE
Type: investmentCodeType

goalCode
Optional. The business goal of the investment.

Table and Column: INV_INVESTMENTS.GOAL_CODE
Type: investmentCodeType

alignment
Optional. The numeric indicator of alignment with business goals between 0 and 100.

Table and Column: INV_INVESTMENTS.ALIGNMENT
Type: Number

risk
Optional. The risk associated with the investment.

Table and Column: INV_INVESTMENTS.RISK
Type: Number

statusIndicator
Optional. The graphical indicator of the investment status.

Table and Column: INV_INVESTMENTS.STATUS_INDICATOR
Type: Number

statusComment
Optional. The text description of investment status.

Table and Column: INV_INVESTMENTS.STATUS_COMMENT
Type: String
progress
The numeric code for investment progress (Not Started, Started, or Completed).
**Default:** 0 - Not Started.
**Table and Column:** INV_INVESTMENTS.PROGRESS
**Type:** Number

**currencyISOcode**
Optional. The ISO code for the currency associated with the investment.
**Table and Column:** INV_INVESTMENTS.CURRENCY_CODE
**Type:** String

**openForTimeEntry**
Optional.
**Table and Column:** INV_INVESTMENTS.IS_OPEN_FOR_TE
**Type:** Boolean

**start**
Optional.
**Table and Column:** INV_INVESTMENTS.SCHEDULE_START
**Type:** investmentDateTimeType

**finish**
Optional.
**Table and Column:** INV_INVESTMENTS.SCHEDULE_FINISH
**Type:** investmentDateTimeType

**cbkType**
Optional. Refers to PRTIMEENTRY_USERLOV1.
**Table and Column:** INV_INVESTMENTS.CBK_TYPE
**Type:** String

**entityCode**
Optional.
**Table and Column:** INV_INVESTMENTS.ENTITY_CODE
**Type:** String
BillExpenseType
Optional. It is either CAPITAL_EXPENDITURE or DEPRECIATION.
Table and Column: INV_INVESTMENTS.BILL_EXPENSE_TYPE
Type: String

trackMode
Optional.
Table and Column: INV_INVESTMENTS.TRACK_MODE
Type: Number

chargeCodeID
Optional.
Table and Column: INV_INVESTMENTS.CHARGECODEID
Type: String

requiredForScenarios
Optional.
Type: Boolean

plannedCostStart
Optional.
Table and Column: FIN_FINANCIALS.PLANNED_CST_START
Type: investmentDateTimeType

plannedCostFinish
Optional.
Table and Column: FIN_FINANCIALS.PLANNED_CST_FINISH
Type: investmentDateTimeType

plannedBenTotal
Optional.
Table and Column: FIN_FINANCIALS.PLANNED_BEN_TOTAL
Type: nonNegativeDouble

plannedBenStart
Optional.
Table and Column: FIN_FINANCIALS.PLANNED_BEN_START
Type: investmentDateTimeType
plannedBenFinish
Optional.
Table and Column: FIN_FINANCIALS.PLANNED_BEN_FINISH
Type: investmentDateTimeType

budgetCostTotal
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_CST_TOTAL
Type: nonNegativeDouble

budgetCostStart
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_CST_START
Type: investmentDateTimeType

budgetCostFinish
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_CST_FINISH
Type: investmentDateTimeType

budgetCostOnHold
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_CST_ONHOLD
Type: investmentDateTimeType

budgetCostResumed
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_CST_RESUMED
Type: investmentDateTimeType

budgetRevTotal
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_REV_TOTAL
Type: nonNegativeDouble

budgetRevStart
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_REV_START
Type: investmentDateTimeType
budgetRevFinish
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_REV_FINISH
Type: investmentDateTimeType

budgetRevOnHold
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_REV_ONHOLD
Type: investmentDateTimeType

budgetRevResumed
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_REV_RESUMED
Type: investmentDateTimeType

forecastCostTotal
Optional.
Table and Column: FIN_FINANCIALS.FORECAST_CST_TOTAL
Type: nonNegativeDouble

forecastCostStart
Optional.
Table and Column: FIN_FINANCIALS.FORECAST_CST_START
Type: investmentDateTimeType

forecastCostFinish
Optional.
Table and Column: FIN_FINANCIALS.FORECAST_CST_FINISH
Type: investmentDateTimeType

forecastCostOnHold
Optional.
Table and Column: FIN_FINANCIALS.FORECAST_CST_ONHOLD
Type: investmentDateTimeType

forecastCostResumed
Optional.
Table and Column: FIN_FINANCIALS.FORECAST_CST_RESUMED
Type: investmentDateTimeType
forecastRevTotal

Optional.
Table and Column: FIN_FINANCIALS.FORECAST_REV_TOTAL
Type: nonNegativeDouble

forecastRevStart

Optional.
Table and Column: FIN_FINANCIALS.FORECAST_REV_START
Type: investmentDateTimeType

forecastRevFinish

Optional.
Table and Column: FIN_FINANCIALS.FORECAST_REV_FINISH
Type: investmentDateTimeType

forecastRevOnHold

Optional.
Table and Column: FIN_FINANCIALS.FORECAST_REV_ONHOLD
Type: investmentDateTimeType

forecastRevResumed

Optional.
Table and Column: FIN_FINANCIALS.FORECAST_REV_RESUMED
Type: investmentDateTimeType

plannedNPV

Optional.
Table and Column: FIN_FINANCIALS.PLANNED_NPV
Type: Number

plannedROI

Optional.
Table and Column: FIN_FINANCIALS.PLANNED_ROI
Type: Number

plannedBreakEven

Optional.
Table and Column: FIN_FINANCIALS.PLANNED_BREAKEVEN
Type: investmentDateTimeType
plannedIRR
  Optional.
  **Table and Column:** FIN_FINANCIALS.PLANNED_IRR
  **Type:** Number

plannedMIRR
  Optional.
  **Table and Column:** FIN_FINANCIALS.PLANNED_MIRR
  **Type:** Number

plannedPaybackPeriod
  Optional.
  **Table and Column:** FIN_FINANCIALS.PLANNED_PAYBACK_PERIOD
  **Type:** string

budgetNPV
  Optional.
  **Table and Column:** FIN_FINANCIALS.BUDGET_NPV
  **Type:** Number

budgetROI
  Optional.
  **Table and Column:** FIN_FINANCIALS.BUDGET_ROI
  **Type:** Number

budgetBreakEven
  Optional.
  **Table and Column:** FIN_FINANCIALS.BUDGET_BREAKEVEN
  **Type:** investmentDateTimeType

budgetIRR
  Optional.
  **Table and Column:** FIN_FINANCIALS.BUDGET_IRR
  **Type:** Number

budgetMIRR
  Optional.
  **Table and Column:** FIN_FINANCIALS.BUDGET_MIRR
  **Type:** Number
budgetPaybackPeriod
Optional.
**Table and Column:** FIN_FINANCIALS.BUDGET_PAYBACK_PERIOD
**Type:** String

forecastNPV
Optional.
**Table and Column:** FIN_FINANCIALS.FORECAST_NPV
**Type:** Number

forecastROI
Optional.
**Table and Column:** FIN_FINANCIALS.FORECAST_ROI
**Type:** Number

forecastBreakEven
Optional.
**Table and Column:** FIN_FINANCIALS.FORECAST_BREAKEVEN
**Type:** investmentDateTimeType

forecastEqualsBudget
Optional.
**Table and Column:** FIN_FINANCIALS.IS_FCST_EQ_BDGT
**Type:** Boolean

calculatePresentValueInfo
Optional.
**Table and Column:** FIN_FINANCIALS.IS_CALC_PV_INFO
**Type:** Boolean

financialLocation
Optional.
**Table and Column:** PAC_MNT_PROJECTS.LOCATIONID
**Type:** String

financialDepartment
Optional.
**Table and Column:** PAC_MNT_PROJECTS.DEPARTCODE
**Type:** String
financialProjectClass
  Optional.
  **Table and Column:** PAC_MNT_PROJECTS.CLASS
  **Type:** String

financialWipClass
  Optional.
  **Table and Column:** PAC_MNT_PROJECTS.WIPCLASS
  **Type:** String

laborRateSource
  Optional.
  **Table and Column:** PAC_MNT_PROJECTS.TRANSRATESOURCELABOR
  **Type:** String

laborCostSource
  Optional.
  **Table and Column:** PAC_MNT_PROJECTS.TRANCOSTSOURCELABOR
  **Type:** String

laborExchangeRateType
  Optional.
  **Table and Column:** PAC_MNT_PROJECTS.LABOR_EXCHANGE_RATE_TYPE
  **Type:** String

materialRateSource
  Optional.
  **Table and Column:** PAC_MNT_PROJECTS.TRANSRATESOURCEMATERIALS
  **Type:** String

materialCostSource
  Optional.
  **Table and Column:** PAC_MNT_PROJECTS.TRANCOSTSOURCEMATERIALS
  **Type:** String

materialExchangeRateType
  Optional.
  **Table and Column:** PAC_MNT_PROJECTS.MATERIALS_EXCHANGE_RATE_TYPE
  **Type:** String
equipmentRateSource
Optional.
Table and Column: PAC_MNT_PROJECTS.TRANSRATESOURCEEQUIPMENT
Type: String

equipmentCostSource
Optional.
Table and Column: PAC_MNT_PROJECTS.TRANSCOSTSOURCEEQUIPMENT
Type: String

equipmentExchangeRateType
Optional.
Table and Column: PAC_MNT_PROJECTS.EQUIPMENT_EXCHANGE_RATE_TYPE
Type: String

expenseRateSource
Optional.
Table and Column: PAC_MNT_PROJECTS.TRANSRATESOURCEEXPENSES
Type: String

expenseCostSource
Optional.
Table and Column: PAC_MNT_PROJECTS.TRANSCOSTSOURCEEXPENSES
Type: String

expenseExchangeRateType
Optional.
Table and Column: PAC_MNT_PROJECTS.EXPENSE_EXCHANGE_RATE_TYPE
Type: String

plannedCostOperatingTotal
Optional. Refers to planned operating cost of an investment.
Table and Column: FIN_FINANCIALS.PLANNED_CST_OPERATING_TOTAL
Type: Double
plannedCostCapitalTotal

Optional. Refers to planned capital cost of an investment.

**Table and Column**: FIN_FINANCIALS.PLANNED_CST_CAPITAL_TOTAL

**Type**: Double

budgetCostOperatingTotal

Optional. Refers to budget operating cost of an investment.

**Table and Column**: FIN_FINANCIALS.BUDGET_CST_OPERATING_TOTAL

**Type**: Double

budgetCostCapitalTotal

Optional. Refers to budget capital cost of an investment.

**Table and Column**: FIN_FINANCIALS.BUDGET_CST_CAPITAL_TOTAL

**Type**: Double

---

**Allocations Schema Tag**

This tag is part of the schema mapping for the financial transaction XOG object.

The Allocations element consists of zero or more allocations. Each Allocation can contain zero or more Detail objects and zero or more CustomInformation objects. The Allocations element at this level indicates the financial allocations used to track charges against different organizational units (for example, a department). The attributes of this tag do not reference tables or columns.

The Allocations schema tag has the following attributes:

**allocationCode**

Required.

**Type**: String

**entityCode**

Optional.

**Type**: String

**locationCode**

Optional.

**Type**: String
**departmentCode**
Optional.
*Type: String*

**resourceClassCode**
Optional.
*Type: String*

**chargeCode**
Optional.
*Type: String*

**investmentCode**
Optional.
*Type: String*

**tableName**
Optional.
*Type: String*

**utilityCode1**
Optional.
*Type: String*

**utilityCode2**
Optional.
*Type: String*

**transactionClassCode**
Optional.
*Type: String*

**transactionClassCode**
Optional.
*Type: String*

**statusCode**
Optional. The status of the allocation.
*Type: String*
Details Schema Tag

This tag is part of the schema mapping for the Financial Transaction XOG object.

The Details element consists of zero or more allocations. Each Detail object can contain zero or more CustomInformation objects. The Detail object can contain a TSV curve consisting of a percentage. The attributes of this tag does not reference tables or columns.

The Details schema tag has the following attributes:

- **glAccountMain**
  - Required.
  - **Type**: String

- **glAccountSub**
  - Required.
  - **Type**: String

- **department**
  - Required.
  - **Type**: String

- **flatAmount**
  - Optional.
  - **Type**: Number

- **weightable**
  - Optional.
  - **Type**: Number
Financial Transaction (scenarioDependencies) Schema Tag

This tag is part of the schema mapping for the Financial Transaction XOG object.

The scenarioDependencies element consists of zero or more scenarioDependency elements. This tag reads data from CA Clarity and confirms that the dependency exists when writing to CA Clarity. If the dependency does not exist, a warning is logged.

The scenarioDependencies schema tag has the following attributes:

**objectInstanceCode**
 Required.

*Table and Column:* INV_INVESTMENTS.CODE
*Type:* String

**name**
 Optional.

*Table and Column:* INV_INVESTMENTS.NAME
*Type:* String

**objectType**
 Optional.

*Table and Column:* ODF_CA_INV.NAME
*Type:* investmentObjectCodeType

**delete**
 Optional.

*Table and Column:* not applicable
*Type:* Boolean
InvestmentAssociations Schema Tag

This tag is part of the schema mapping for the Financial Transaction XOG object.

The InvestmentAssociations element consists of zero or more Allocations and Hierarchies elements. Allocations refers to the parent investments of the current investment and Hierarchies to the child investments of the current investments.

Allocations

The Allocations element at this level determines what percentage of the cost of an investment is to be included in any parent investment (for example, Oracle may be used by three different applications so the cost of Oracle can be split between the three applications). The allocations element is a wrapper for zero or more ParentInvestment elements.

ParentInvestment

The ParentInvestment element can contain the following attributes, in addition to zero or more CustomInformation elements:

InvestmentID

Required. The ID of the immediate parent of the current investment.

**Table and Column:** INV_HIERARCHIES.PARENT_ID

**Type:** investmentIDType

InvestmentType

Required.

**Table and Column:** None

**Type:** investmentObjectCodeType

defaultAllocationPercent

Required. The percentage of the child budget and staff cost to allocate to the parent. This attribute must total to either "0" or "1." This is because an investment must be either completely allocated to its parents or completely unallocated.

**Table and Column:** INV_HIERARCHIES.DEFAULT_ALLOC_PCT

**Type:** Number
Hierarchies

The Hierarchies element is a wrapper for zero or more ChildInvestment elements.

ChildInvestment

The ChildInvestment element can contain zero or more CustomInformation elements and the following attributes:

InvestmentID

Required. The ID of the immediate child of the current investment.

Table and Column: INV_HIERARCHIES.CHILD_ID
Type: investmentIDType

InvestmentType

Required. The type of investment.

Table and Column: None
Type: investmentObjectCodeType

defaultAllocationPercent

Optional. The percentage of the child budget and staff cost to allocate to the parent.

Table and Column: INV_HIERARCHIES.DEFAULT_ALLOC_PCT
Type: Number

InvestmentBaselines Schema Tag

This tag is part of the schema mapping for the Financial Transaction XOG object.

The InvestmentBaselines schema tag contains the following attributes including zero or more InvestmentBaseline elements. Each InvestmentBaseline element can contain a BaselineDetails element.

revisionID

Required.

Table and Column: PRJ_BASELINES.CODE
Type: String

delete

Optional.

Table and Column: None
Type: Boolean
**currentRevision**

Required.

*Table and Column*: PRJ_BASELINES.IS_CURRENT

*Type*: Boolean

**name**

Required.

*Table and Column*: PRJ_BASELINES.NAME

*Type*: String

**description**

Optional.

*Table and Column*: PRJ_BASELINES.DESCRIPTION

*Type*: String

**BaselineDetails**

The BaselineDetails element contains detailed usage or cost curves for a baseline.

**start**

Optional.

*Table and Column*: PRJ_BASELINES_DETAILS.START_DATE

*Type*: projectDateTimeType

**finish**

Optional.

*Table and Column*: PRJ_BASELINES_DETAILS.FINISH_DATE

*Type*: projectDateTimeType

**usageSum**

Optional.

*Table and Column*: PRJ_BASELINES_DETAILS.USAGE_SUM

*Type*: Number

**costSum**

Optional.

*Table and Column*: PRJ_BASELINES_DETAILS.COST_SUM

*Type*: Number
duration

Optional.

**Table and Column:** PRJ_BASELINESDETAILS.DURATION

**Type:** Number

**UsageCurve and CostCurve Schema Tags**

These tags are part of the schema mapping for the Financial Transaction XOG object. The curve elements contain segment objects which specify cost or usage over a period of time. The UsageCurve and CostCurve schema tags have the following attributes:

**start**

Required.

**Table and Column:** None

**Type:** projectDateTimeType

**finish**

Required.

**Table and Column:** None

**Type:** projectDateTimeType

**sum**

Required.

**Table and Column:** None

**Type:** Number

**InvestmentResources Schema Tag**

This tag is part of the schema mapping for the Financial Transaction XOG object. The InvestmentResources element contains zero or more Resource elements that make up the Team element for an investment. This tag includes the following attributes:

**Resource**

**availFrom**

Optional. If the Team field is not set, use the investment start.

**Table and Column:** PRTEAM.pravailstart

**Type:** Date
**availTo**
Optional. If the team field is not set, use the investment finish.
*Table and Column:* PRTEAM.pravailfinish
*Type:* Date

**openForTimeEntry**
Optional.
*Table and Column:* PRTEAM.prisopen
*Type:* Boolean

**bookingStatus**
Optional.
*Table and Column:* PRTEAM.prbooking
*Type:* Integer

**requestStatus**
Optional.
*Table and Column:* PRTEAM.prstatus
*Type:* Integer

**defaultAllocation**
Optional.
*Table and Column:* PRTEAM.pralloccurve
*Type:* Float

**resourceID**
Required.
*Table and Column:* PRTEAM.prresourceid
*Type:* String

**projectRoleID**
Optional.
*Table and Column:* PRTEAM.prroleid
*Type:* String

**isProjectManager**
Optional. Indicates if the resource is the project manager.
*Table and Column:* Not applicable
*Type:* Boolean
lastUpdatedBy
  Optional.
  Table and Column: PRTEAM.last_updated_by
  Type: String

lastUpdatedDate
  Optional.
  Table and Column: PRTEAM.last_updated_date
  Type: Date

teamId
  Optional. The unique identifier for each team member on an investment.
  Table and Column: PRTEAM.team_uid
  Type: String

requirementName
  Optional.
  Table and Column: PRTEAM.requirement_name
  Type: String

Subelement <Baselines>

A subelement for Resource or Task. It has the following attributes:

revisionID
  Required.
  Table and Column: PRJ_BASELINES.code
  Type: String

costSum
  Optional.
  Table and Column: PRJ_BASELINE_DETAILS.cost_sum
  Type: Float

duration
  Optional.
  Table and Column: PRJ_BASELINE_DETAILS.duration
  Type: Float
start  
Optional.  
**Table and Column:** PRJ_BASELINE_DETAILS.start_date  
**Type:** Date

finish  
Optional.  
**Table and Column:** PRJ_BASELINE_DETAILS.finish_date  
**Type:** Date

**usageSum**  
Optional.  
**Table and Column:** PRJ_BASELINE_DETAILS.usage_sum  
**Type:** Float

**Subelement `<AllocCurve>`**

AllocCurve is a sub element of the Resource element.  

**AllocCurve**  
**Table and Column:** PRTEAM.prAllocCurve  
**Type:** Varied

**Subelement `<HardAllocCurve>`**

HardAllocCurve is a sub element of the Resource element.  

**HardAllocCurve**  
**Table and Column:** PRTEAM.prAllocCurve  
**Type:** Varied

**Subelement `<SkillAssocs>`**

SkillAssocs is a sub element of the Resource element.  

**SkillAssocs**  
A subelement of the Resource element.
**InvestmentTasks Schema Tag**

This tag is part of the schema mapping for the Financial Transaction XOG object. The InvestmentTasks element contains zero or more Task elements.

**Task**

The attributes of the task element are not associated with any table or column. The Task element has the following attributes:

- **internalTaskID**
  - Optional.
  - Type: String

- **delete**
  - Optional.
  - Type: Boolean

- **taskID**
  - Optional.
  - Type: String

- **name**
  - Optional.
  - Type: String

- **parent**
  - Optional.
  - Type: String

- **firstChildOf**
  - Optional.
  - Type: String

- **lastChildOf**
  - Optional.
  - **Table and Column**: Not applicable
  - Type: String

- **nestSiblingOf**
  - Optional.
  - Type: String
**orderID**
Optional.
*Type: Number*

**outlineLevel**
Optional.
*Type: Number*

**start**
Optional.
*Type: projectDateTimeType*

**baseStart**
Optional.
*Type: projectDateTimeType*

**finish**
Optional.
*Type: projectDateTimeType*

**baseFinish**
Optional.
*Type: projectDateTimeType*

**milestone**
Optional.
*Type: Boolean*

**summary**
Optional.
*Type: Boolean*

**key**
Optional.
*Type: Boolean*

**category**
Optional.
*Type: String*
status
  Optional.
  Type: Number

percComp
  Optional.
  Type: Number

lastUpdatedBy
  Optional.
  Type: String

lastUpdatedDate
  Optional.
  Type: projectDateTimeType

priority
  Optional.
  Type: Number

earlyStart
  Optional.
  Type: projectDateTimeType

lateStart
  Optional.
  Type: projectDateTimeType

earlyFinish
  Optional.
  Type: projectDateTimeType

lateFinish
  Optional.
  Type: projectDateTimeType

duration
  Optional.
  Type: Number
baselineDuration
  Optional.
  Type: Number

totalSlack
  Optional.
  Type: Number

unplanned
  Optional.
  Type: Boolean

shortName
  Optional.
  Type: String

guidelines
  Optional.
  Type: String

fixed
  Optional.
  Type: Boolean

lockedForScheduling
  Optional.
  Type: Boolean

baseFixed
  Optional.
  Type: Boolean

baseTime
  Optional.
  Type: projectDateTimeType

critical
  Optional.
  Type: Boolean
chargeCodeID
  Optional.
  Type: Number

subproject
  Optional.
  Type: Boolean

subprojectID
  Optional.
  Type: String

subprojectTaskID
  Optional.
  Type: String

subprojectReadOnly
  Optional.
  Type: Boolean

subprojectIPD
  Optional.
  Type: Boolean

userText1
  Optional.
  Type: String

topDownPercent
  Optional.
  Type: Number

chargeCodeExtID
  Optional.
  Type: String
General Schema Tag

This tag is part of the schema mapping for the Financial Transaction XOG object. This tag is not associated with tables and columns. It has the following attributes:

**addedBy**
- Optional.
- **Type:** String

**addedDate**
- Optional.
- **Type:** Date

OBSAssocs Schema Tag

This tag is part of the schema mapping for the Financial Transaction XOG object. The OBSAssocs element contains the following element, plus zero or more OBSAssoc subelements.

**completed**
- Optional. When completed and this value is set to True. Existing OBS associations not listed in the import are deleted.
- **Default:** False.
- **Table and Column:** None
- **Type:** Boolean

OBSAssoc Element

No tables or columns are associated with the OBSAssoc element. It has the following attributes:

**name**
- Optional.
- **Type:** String

**id**
- Optional.
- **Type:** String

**unitPath**
- Required. This is a slash-delimited list of unit names leading to the unit to which the object is associated (for example, "CAN/BC/VAN").
- **Type:** String
mode

Optional.

Type: OBSRightMode

Custom Information Schema Tag

This tag is part of the schema mapping for the Financial Transaction XOG object.

The CustomInformation element contains data for custom attributes added to the Investment object.
Issue

Use the issue XOG object to view inbound and outbound issue instances.

Schema Name

nikuxog_issue.xsd

Read and Write XML Files

The following XML files are included:
- issue_read.xml. Use this file to export issue instances from CA Clarity.
- issue_write.xml. Use this file to import issue instances that were previously exported from CA Clarity.

Prerequisites

Before using this XOG object, make sure the referenced objects, such as the project, user, and category, exist in CA Clarity.

Read Filters

The following explicit read filters are used:

- **projectCode**
  Defines the code for the associated project.

- **Name**
  Defines the name of the issue.

- **riskCode**
  Defines the risk of the issue.

- **statusCode**
  Defines the status of the issue.

- **priorityCode**
  Defines the priority of the issue.

- **ownerCode**
  Defines the name of the owner or assignee of the issue.

Error Handling

The following errors can be thrown:
- Project does not exist in the system.
■ Category type is not valid.
■ Status is not valid.
■ Priority is not valid.
■ Owner does not exist in the system.
■ Impact is not valid.
■ Resolved By does not exist in the system.
■ Failed to import risk/issue/change request.

**Schema Mapping**

Mappings for the following schema tag names are provided:

■ Issue
**Issue Schema Tag**

The issue tag is part of the schema mapping for the issue XOG object. It has the following attributes:

- **ownerCode**
  - Required. Defines the name of the resource assigned to this issue.
  - **Table and Column:** ASSIGNED_TO
  - **Type:** String

- **categoryTypeCode**
  - Defines the category of this issue.
  - **Table and Column:** CATEGORY_TYPE_CODE
  - **Type:** String

- **description**
  - Defines the description of this issue.
  - **Table and Column:** DESCRIPTION
  - **Type:** String

- **code**
  - Required. Defines the unique identifier for this issue.
  - **Table and Column:** RIM_RISK_ISSUE_CODE
  - **Type:** String

- **impactDate**
  - Defines the impact date for this issue.
  - **Table and Column:** IMPACT_DATE
  - **Type:** Date

- **name**
  - Required. Defines the name of the issue.
  - **Table and Column:** NAME
  - **Type:** String

- **priorityCode**
  - Defines the priority of this issue.
  - **Table and Column:** PRIORITY_CODE
  - **Type:** String

- **projectCode**
  - Required. Defines the project associated with this issue.
Table and Column: INV_INVESTMENTS.CODE
Type: String

statusCode
Defines the status of this issue.
Table and Column: STATUS_CODE
Type: String

resolution
Defines the description of the resolution for this issue.
Table and Column: RESOLUTION
Type: String

resolvedBy
Defines the name of the resource who resolved the issue.
Table and Column: RESOLVED_BY
Type: Number

resolvedDate
Defines the date the issue was resolved.
Table and Column: RESOLVED_DATE
Type: Date

targetResolutionDate
Defines the date the issues is expected to be resolved and closed.
Table and Column: TARGET_RESOLUTION_DATE
Type: Date
Location

Use the location XOG object to view inbound and outbound location attributes.

Schema Name

nikuxog_location.xsd

Read and Write XML Files

The following XML files are included:

- locations_read.xml. Use this file to export locations from CA Clarity.
- locations_write.xml. Use this file to import locations that were previously exported from CA Clarity.

Prerequisites

An entity must exist in CA Clarity.

Business Rules and Processing

When a location is created, a corresponding OBS unit is created in the location OBS referred to by the location's entity.

Read Filters

The following explicit read filter is used:

Entity

The unique entity code for which the locations should be read out.

Error Handling

The following attribute values are validated against CA Clarity. If the values do not exist, XOG displays an error message and does not import or update the record.

Entity

Checks if the entity is valid and exists in the system.

Required fields

Ensures all required fields have values.

Department associations

Ensures departments belong to the same entity. Further, if the department does not exist, a warning is output.
Schema Mappings

The following schema mappings are provided for locations:

- Locations
- Description
- DepartmentAssociations
- Child Location

Locations Schema Tag

The Locations tag is part of the schema mapping for the Location XOG object.

A placeholder element for multiple locations. This schema tag has the following elements:

- Location
- Description Schema Tag
- DepartmentAssociations Schema Tag

Location Element

The Location element is the actual Location object. It has the following attributes.

locationCode

Required. The unique code for location.

**Table and Column:** locations.locationid

**Type:** String

entity

Required. The entity to which the location belongs.

**Table and Column:** locations.entity

**Type:** String

description

Required.

**Table and Column:** locations.locationdescription

**Type:** String
shortdescription
   Required.
   Table and Column: locations.shortdesc
   Type: String
address1
   Optional.
   Table and Column: locations.address1
   Type: String
address2
   Optional.
   Table and Column: locations.address2
   Type: String
address3
   Optional.
   Table and Column: locations.address3
   Type: String
city
   Optional.
   Table and Column: locations.city
   Type: String
zip
   Optional.
   Table and Column: locations.zip
   Type: String
countryid
   Optional.
   Table and Column: locations.countryid
   Type: String
phone
   Optional.
   Table and Column: locations.phone
   Type: String
fax
Optional.
**Table and Column:** locations.fax
**Type:** String

managerResourceCode
Optional. The manager for location department associations.
**Table and Column:** locations.manager_resource_code
**Type:** String

name
Required. The name of the location.
**Table and Column:** locations.shortdesc
**Type:** String

locationManagerCode
Optional. The resource code for location manager.
**Table and Column:** locations.location_manager_id
**Type:** String

stateprov
Optional.
**Table and Column:** locations.stateprov
**Type:** String

**Description Schema Tag**

The Description schema tag contains the attribute for the location description.

Description
Required. The description of the location.
**Table and Column:** description
**Type:** String
DepartmentAssociations Schema Tag

The placeholder element for multiple department associations. It has the following attribute.

DepartmentAssociation

Represents the department that is associated to the location. It has the following attribute:

departmentCode

Required. The association to department code. The department must belong to the same entity as the location.

Table and Column: LocationDept.dept_id

Type: String

Child Location Schema Tag

A child location. This element contains all the elements and attributes that the enclosing location has.
Matrix

Use the matrix XOG object to view inbound and outbound cost/rate matrix instances. Rate matrices give you flexibility in defining cost for particular services or resources.

Schema Name

nikuxog_matrix.xsd

Read and Write XML Files

The following XML files are included:
- matrices_read.xml. Use this file to export matrices from CA Clarity.
- matrices_write.xml. Use this file to import matrices that were previously exported from CA Clarity.

Filters

The following explicit read filters are used:
- name
  - The name of the matrix.
- type
  - The type of the matrix (Cost or Rate).
- location
  - The location of the matrix.

Error Handling

The following errors can be thrown when importing or exporting the cost plan:
- Failed to export matrices.
- Failed to import matrices.

Schema Mappings

Mappings for the following schema tag names are provided:
- columnType (see page 290)
- MatrixRowType (see page 292)
**columnType Schema Tag**

This tag is part of the schema mapping for the matrix XOG object. The columns you assign to a matrix through the XOG are saved to the PPA_MATRIXCOLDEF table.

This tag has the following attribute:

**name**

Optional. Defines the list of columns to be added to the matrix rows. It has the following attributes:

**chargeCode**

Defines the charge code for the matrix column.

*Table and Column:* PPA_MATRIXCOLDEF.FIELDNAME

*Type:* String

**clntclass**

Optional. Defines the client class for the matrix column.

*Table and Column:* PPA_MATRIXCOLDEF.FIELDNAME

*Type:* String

**company_code**

Defines the company code for the matrix row.

*Table and Column:* PPA_MATRIXCOLDEF.FIELDNAME

*Type:* String

**departcode**

Defines the department code for the matrix column.

*Table and Column:* PPA_MATRIXCOLDEF.FIELDNAME

*Type:* String

**entity**

Defines the entity for the matrix column.

*Table and Column:* PPA_MATRIXCOLDEF.FIELDNAME

*Type:* String

**inputtype**

Defines the input type code for the matrix column.

*Table and Column:* PPA_MATRIXCOLDEF.FIELDNAME

*Type:* String
locationid
Defines the location unique identifier for the matrix column.
**Table and Column:** PPA_MATRIXCOLDEF.FIELDNAME
**Type:** String

projclass
Defines the project class for the matrix column.
**Table and Column:** PPA_MATRIXCOLDEF.FIELDNAME
**Type:** String

project_code
Defines the project code for the matrix column.
**Table and Column:** PPA_MATRIXCOLDEF.FIELDNAME
**Type:** String

resourceClass
Defines the resource class for the matrix column.
**Table and Column:** PPA_MATRIXCOLDEF.FIELDNAME
**Type:** String

resource_code
Defines the resource code for the matrix column.
**Table and Column:** PPA_MATRIXCOLDEF.FIELDNAME
**Type:** String

resourceRole
Optional. Defines the resource role for the matrix column.
**Table and Column:** PPA_MATRIXCOLDEF.FIELDNAME
**Type:** String

transactionClass
Optional. Defines the transaction class for the matrix column.
**Table and Column:** PPA_MATRIXCOLDEF.FIELDNAME
**Type:** String

projsitecode
Defines the project site code for the matrix column.
**Table and Column:** PPA_MATRIXCOLDEF.FIELDNAME
**Type:** String
MatrixRowType Schema Tag

The MatrixRowType tag is part of the schema mapping for the matrix XOG object. The values for the rows you add to the matrix are saved to a column in the PPA_MATRIXVALUES table.

This tag has the following attributes:

**fromDate**

Optional. Defines the date from which to apply the matrix row.

*Table and Column:* PPA_MATRIXVALUES.FROMDATE

*Type:* Date

**toDate**

Optional. Defines the date until which to apply the matrix row.

*Table and Column:* PPA_MATRIXVALUES.TODATE

*Type:* Date

**chargecode**

Optional. Defines the chargecode for the matrix row.

*Table and Column:* PPA_MATRIXVALUES.VALUE1

*Type:* String

**Department**

Optional. Defines the department for the matrix row.

*Table and Column:* PPA_MATRIXVALUES.VALUE2

*Type:* String

**entity**

Optional. Defines the entity for the matrix row.

*Table and Column:* PPA_MATRIXVALUES.VALUE3

*Type:* String

**Input type code**

Optional. Defines the input type code for the matrix row.

*Table and Column:* PPA_MATRIXVALUES.VALUE6

*Type:* String
location
Optional. Defines the location for the matrix row.

Table and Column: PPA_MATRIXVALUES.VALUE9
Type: String

Project class
Optional. Defines the project class for the matrix row.

Table and Column: PPA_MATRIXVALUES.VALUE8
Type: String

Project
Optional. Defines the project for the matrix row.

Table and Column: PPA_MATRIXVALUES.VALUE7
Type: String

Resource class
Optional. Defines the resource class for the matrix row.

Table and Column: PPA_MATRIXVALUES.VALUE9
Type: String

Resource
Optional. Defines the resource for the matrix row.

Table and Column: PPA_MATRIXVALUES.VALUE10
Type: String

rate
Optional. Defines the rate for the matrix row.

Table and Column: PPA_MATRIXVALUES.NUMVAL1
Type: Double

standardCost
Optional. Defines the standard cost for the matrix row.

Table and Column: PPA_MATRIXVALUES.NUMVAL2
Type: Double

actualCost
Optional. Defines the actual cost for the matrix row.

Table and Column: PPA_MATRIXVALUES.NUMVAL3
Type: Double
currencyCode

Optional. Defines the currency code for the matrix row.

**Table and Column:** PPA_MATRIXVALUES.MATRIX_CURRENCY_CODE

**Type:** String

typeCode

Optional. Defines the cost plus code for the matrix row.

**Table and Column:** PPA_MATRIXVALUES.STRVAL1

**Type:** String
Non-Project Investment

Use the Non-project Investments XOG object to view inbound and outbound non-project investment object (NPIO) attributes. NPIOs include:

- Asset
- Application
- Product
- Other Work

Schema Names

The following schema files are part of this XOG object:

- nikuxog_asset.xsd
- nikuxog_application.xsd
- nikuxog_product.xsd
- nikuxog_otherInvestment.xsd

Read and Write XML Files

None

Business Rules and Processing

The following business rules and processing apply to this XOG:

- The non-project investments are flat objects with no hierarchical data.
- Each object contains many of the same fields and has essentially the same behavior.
- NPIOs are defined for both inbound (write) and outbound (read) processing. On import, if the NPIO exists, it is posted. CA Clarity checks to ensure that the object type and name matches. If the NPIO does not exist, CA Clarity checks to ensure that the name is unique. If the check passes, the NPIO is created and posted.
- This XOG imports and exports the team members, if the NPIO has been staffed. The resources must exist on the target system during an import or the import XOG will fail if the NPIO has team members.

Read Filters

The XOG supports the outbound processing of NPIOs based on the following fields:

**ObjectId**

The unique ID for the NPIO.

**managerUserName**
The name of the NPIO manager.

**lastUpdatedDate**

The last date when the NPIO was updated.

If all filters are commented out, all NPIOs for the defined type are exported.

**Terms**

The following terms are used in this section:

**managerUserName**

The name of the NPIO manager.

The manager is validated against the CMN_SEC_USERS.USER_NAME field. If the manager does not exist, the NPIO is imported without a manager and a warning is posted to the Success and Error file.

**approvedByID**

The unique identifier for the NPIO approver.

The approver is validated against the CMN_SEC_USERS.USER_NAME field. If the approver does not exist, the NPIO is not imported and a warning is posted to the Success and Error file.

**objectID**

The unique identifier for the object.

If the objectID:

- Exists and it matches the ID on a different object type, the object is not imported and an error is posted to the Success and Error file.
- Does not exist and auto-numbering is not active for the object type, the object is not imported and an error is posted to the Success and Error file.

**chargeCodeExtID**

The charge code associated to the NPIO.

The charge code is validated against the PRCHARGE_CODE table. If the charge code does not exist, the NPIO is not imported and an error is posted to the Success and Error file.

**goalCode**

The goal code associated with an NPIO.

This is validated against the CMN_LOOKUP_TYPE/CMN_LOOKUPS table. If the charge code does not exist, the NPIO is not imported and an error is posted to the Success and Error file.

**processCode**

The process code associated with an NPIO.
This is validated against the CMN_LOOKUP_TYPE/CMN_LOOKUPS table. If the process code does not exist, the NPIO is not imported and an error is posted to the Success and Error file.

**stageCode**

The stage code associated with an NPIO.

The stage code is validated against the CMN_LOOKUP_TYPE/CMN_LOOKUPS table. If the stage code does not exist, the NPIO is not imported and an error is posted to the Success and Error file.

**categoryCode**

The category code associated with an NPIO.

This is validated against the CMN_LOOKUP_TYPE/CMN_LOOKUPS table. If the category code does not exist, the NPIO is not imported and an error is posted to the Success and Error file.

**currencyISOcode**

The currency code associated to the NPIO.

The currency code is validated against the CMN_OPTIONS/CMN_OPTION_VALUES/CMN_CURRENCIES tables. If the currency is not active or cannot be converted into the base currency, the NPIO is not imported and an error is posted to the Success and Error file.

### Schema Mappings

Mappings for the following schema tag names are provided:

- **Asset** (see page 297)
- **Application** (see page 298)
- **Product** (see page 300)
- **Other Work** (see page 300)
- **NPIO Common Fields** (see page 301)

### Asset Schema Tag

This tag is part of the schema mapping for the Non-project Investment XOG object. The asset schema tag has the following attribute:

**totalCostOfOwnership**

Optional. The minimum is 0.

**Table and Column:** PRI_PROJECTS.INV_TCO  
**Type:** Double
Application Schema Tag

This tag is part of the schema mapping for the Non-project Investment XOG object.

The Application schema tag has the following attributes:

**totalCostOfOwnership**
- Optional. The minimum is 0.
- **Table and Column:** PRJ_PROJECTS.INV_TCO
- **Type:** Double

**investmentVersion**
- **Table and Column:** PRJ_PROJECTS.INV_VERSION
- **Type:** String

**supplier**
- **Table and Column:** PRJ_PROJECTS.PRJ_PROJECTS.INV_SUPPLIER
- **Type:** String

**populationServed**
- The minimum is 0.
- **Table and Column:**
- **Type:** Number

**licenseCount**
- The minimum is 0.
- **Table and Column:**
- **Type:** Number

**functionPoints**
- The minimum is 0.
- **Table and Column:**
- **Type:** Number
technology

Defines the technology.

Values:
- 0. data
- 1. desktop
- 2. video
- 3. voice

Table and Column:
Type: Number

platform

Defines the platform.

Values:
- 0. HP-UX
- 1. Macintosh
- 2. Sun
- 3. Windows
- 4. other

Table and Column:
Type: Number
**Product Schema Tag**

The product tag is part of the schema mapping for the Non-project Investment XOG object. The product schema tag has the following attribute:

**investmentVersion**

Optional.

*Table and Column:* PRJ_PROJECTS.INV_VERSION

*Type:* String

**Schema Name**

nikuxog_product.xsd

**Read and Write XML Files**

The following XML files are included:

- `inv_products_read.xml`. Use this file to export product instances from CA Clarity.
- `inv_products_write.xml`. Use this file to import product instances that were previously exported from CA Clarity.

**Other Work Schema Tag**

This tag is part of the schema mapping for the Non-project Investment XOG object. There are no specific attributes for the Other Work schema.

**Schema Name**

nikuxog_otherInvestment.xsd

**Read and Write XML Files**

The following XML files are included:

- `inv_others_read.xml`. Use this file to export other investment instances from CA Clarity.
- `inv_others_write.xml`. Use this file to import other investment instances that were previously exported from CA Clarity.
NPIO Common Fields Schema Tag

The NPIO Common Fields tag is part of the schema mapping for the non-project investment XOG object. This schema tag has the following attributes:

**name**

Required. Defines the name of the investment.

**Table and Column:** SRM_PROJECTS.NAME  
**Type:** String

**objectId**

Required. Defines the unique identifier for the investment. You must provide a unique identifier if CA Clarity is not configured for autonumbering.

**Table and Column:** SRM_PROJECTS.UNIQUE_NAME  
**Type:** String

**status**

Optional. Defines the status of the investment.

**Values:**

- 0. Unapproved
- 1. Approved
- 2. On Hold
- 3. Rejected
- 4. Cancelled
- 5. Resumed

**Table and Column:** PRJ_PROJECTS.STATUS  
**Type:** Number

**description**

Optional.

**Table and Column:** SRM_PROJECTS.DESCRIPTION  
**Type:** String
priority
Optional. Defines the priority of the investment.

Values: 0-36, where:
- 0. Highest priority
- 36. Lowest priority

Table and Column: PRJ_PROJECTS.prPRIORITY
Type: Number

managerUserName
Optional. Defines the name of the resource managing the investment. The manager's user name must match an existing user name in CA Clarity.

Table and Column: PRJ_PROJECTS.MANAGER_ID
Type: String

approvedById
Required. Defines the name of the resource who approved the investment. The user name must match an existing user name in CA Clarity.

Table and Column: PRJ_PROJECTS.APPROVEDBY_ID
Type: String

chargeCodeExtId
Required.

Table and Column: PRJ_PROJECTS.PRCHARGECODEID
Type: String

lastUpdatedBy
Optional. Defines the name of the resource who last updated the investment.

Default: The name of the resource logged in to CA Clarity.

Table and Column: PRJ_PROJECTS.LAST_UPDATED_BY
Type: String

lastUpdatedDate
Optional. Defines the date the investment was last updated.

Default: Today's date

Table and Column: SRMPROJECTS.LAST_UPDATED_DATE
Type: dateTime
**approvedTime**

Optional. Defines the time the investment was approved.

*Table and Column:* PRJ_PROJECTS.PRAPPROVEDTIME  
*Type:* dateTime

**processCode**

Required. Defines the process code of the investment. The code must exist in CA Clarity.

*Table and Column:* PRJ_PROJECTS.PROCESS_CODE  
*Type:* String

**stageCode**

Required. Defines the stage code of the investment. The code must exist in CA Clarity.

*Table and Column:* PRJ_PROJECTS.STAGE_CODE  
*Type:* String

**categoryCode**

Required. Defines the category code of the investment. The code must exist in CA Clarity.

*Table and Column:* PRJ_PROJECTS.CATEGORY_CODE  
*Type:* String

**goalCode**

Required. Defines the goal code of the investment. The code must exist in CA Clarity.

*Table and Column:* PRJ_PROJECTS.GOAL_CODE  
*Type:* String

**alignment**

Optional. Defines the alignment of the investment.

*Table and Column:* PRJ_PROJECTS.ALIGNMENT  
*Type:* Number

**risk**

Optional. Defines the risk of the investment.

*Table and Column:* PRJ_PROJECTS.RISK  
*Type:* Number
statusIndicator
Optional.
Table and Column: PRJ_PROJECTS.STATUS_INDICATOR
Type: Number

statusComment
Optional.
Table and Column: PRJ_PROJECTS.STATUS_COMMENT
Type: String

progress
Optional.
Values:
- 0. Not Started
- 1. Started
- 2. Completed
Table and Column: PRJ_PROJECTS.PROGRESS
Type: Double

budgetCostTotal
Optional. The minimum is 0.
Table and Column: PRJ_PROJECTS.BDGT_CST_TOTAL
Type: dateTime

budgetCostStart
Optional.
Table and Column: PRJ_PROJECTS.BDGT_CST_START
Type: dateTime

budgetCostFinish
Optional.
Table and Column: PRJ_PROJECTS.BDGT_CST_FINISH
Type: dateTime

budgetCostOnHold
Optional.
Table and Column: PRJ_PROJECTS.BDGT_CST_ONHOLD
Type: dateTime
**budgetCostResume**
Optional.

**Table and Column:** PRJ_PROJECTS.BDGT_CST_RESUME  
**Type:** dateTime

**budgetRevTotal**
Optional. The minimum is 0.

**Table and Column:** PRJ_PROJECTS.BDGT_REV_TOTAL  
**Type:** Double

**budgetRevStart**
Optional.

**Table and Column:** PRJ_PROJECTS.BDGT_REV_START  
**Type:** dateTime

**budgetRevFinish**
Optional.

**Table and Column:** PRJ_PROJECTS.BDGT_REV_FINISH  
**Type:** dateTime

**budgetRevOnHold**
Optional.

**Table and Column:** PRJ_PROJECTS.BDGT_REV_ONHOLD  
**Type:** dateTime

**budgetRevResume**
Optional.

**Table and Column:** PRJ_PROJECTS.BDGT_REV_RESUME  
**Type:** dateTime

**budgetNPV**
Optional. If the Calculate Present Value Info flag is set, the budgetNPV is recalculated after import.

**Table and Column:** PRJ_PROJECTS.BDGT_NPV  
**Type:** Float
budgetROI
Optional. If the Calculate Present Value Info flag is set, the budgetROI is recalculated after import. 1.0 is 100%.
Table and Column: PRJ_PROJECTS.BDGT_ROI
Type: Percent

budgetBreakEven
Optional. This is recalculated after import if Calculate Present Value Info flag is set.
Table and Column: PRJ_PROJECTS.BDGT_BREAKEVEN
Type: dateTime

forecastCostTotal
Optional. The minimum is 0.
Table and Column: PRJ_PROJECTS.FCST_CST_TOTAL
Type: Double

forecastCostStart
Optional.
Table and Column: PRJ_PROJECTS.FCST_CST_START
Type: dateTime

forecastCostFinish
Optional.
Table and Column: PRJ_PROJECTS.FCST_CST_FINISH
Type: dateTime

forecastCostOnHold
Optional.
Table and Column: PRJ_PROJECTS.FCST_CST_ONHOLD
Type: dateTime

forecastCostResume
Optional.
Table and Column: PRJ_PROJECTS.FCST_CST_RESUME
Type: dateTime

forecastRevTotal
Optional.
Table and Column: PRJ_PROJECTS.FCST_REV_TOTAL
Type: dateTime
forecastRevStart
Optional.
Table and Column: PRJ_PROJECTS.FCST_REV_START
Type: dateTime

forecastRevFinish
Optional.
Table and Column: PRJ_PROJECTS.FCST_REV_FINISH
Type: dateTime

forecastRevOnHold
Optional.
Table and Column: PRJ_PROJECTS.FCST_REV_ONHOLD
Type: dateTime

forecastRevResume
Optional.
Table and Column: PRJ_PROJECTS.FCST_REV_RESUME
Type: dateTime

forecastNPV
Optional. This is recalculated after import if Calculate Present Value Info flag is set.
Table and Column: PRJ_PROJECTS.FCST_NPV
Type: Float

forecastROI
Optional. This is recalculated after import if Calculate Present Value Info flag is set. 1.0 is 100%.
Table and Column: PRJ_PROJECTS.FCST_ROI
Type: Percent

forecastBreakEven
Optional. This is recalculated after import if Calculate Present Value Info flag is set.
Table and Column: PRJ_PROJECTS.FCST_BREAKEVEN
Type: dateTime
currencyISOcode
Required. The three-character ISO currency code that must be an active currency in CA Clarity.
Table and Column: PRJ_PROJECTS.CURRENCY_CODE
Type: String
calculatePresentValueInfo
Required. If selected, this calculates the NPV, ROI and breakeven after import.
Table and Column: PRJ_PROJECTS.IS_CALC_PV_INFO
Type: Boolean
forecastEqualsBudget
Optional. If selected, forecast values will track the budget values.
Table and Column: PRJ_PROJECTS.IS_CALC_PV_INFO
Type: Boolean
plannedCostStart
Optional.
Table and Column: FIN_FINANCIALS.PLANNED_CST_START
Type: Date
plannedCostFinish
Optional.
Table and Column: FIN_FINANCIALS.PLANNED_CST_FINISH
Type: Date
plannedBenTotal
Optional.
Table and Column: FIN_FINANCIALS.PLANNED_BEN_TOTAL
Type: Double
plannedBenStart
Optional.
Table and Column: FIN_FINANCIALS.PLANNED_BEN_START
Type: Date
plannedBenFinish
Optional.
Table and Column: FIN_FINANCIALS.PLANNED_BEN_FINISH
Type: Date

budgetCostTotal
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_CST_TOTAL
Type: Double

budgetCostStart
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_CST_START
Type: Date

budgetCostFinish
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_CST_FINISH
Type: Date

budgetRevTotal
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_REV_TOTAL
Type: Double

budgetRevStart
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_REV_START
Type: Date

budgetRevFinish
Optional.
Table and Column: FIN_FINANCIALS.BUDGET_REV_FINISH
Type: Date

forecastCostTotal
Optional.
Table and Column: FIN_FINANCIALS.FORECAST_CST_TOTAL
Type: Double
Non-Project Investment

forecastCostStart
Optional.
**Table and Column:** FIN_FINANCIALS.FORECAST_CST_START  
**Type:** Date

forecastCostFinish
Optional.
**Table and Column:** FIN_FINANCIALS.FORECAST_CST_FINISH  
**Type:** Date

forecastRevTotal
Optional.
**Table and Column:** FIN_FINANCIALS.FORECAST_REV_TOTAL  
**Type:** Double

forecastRevStart
Optional.
**Table and Column:** FIN_FINANCIALS.FORECAST_REV_START  
**Type:** Date

forecastRevFinish
Optional.
**Table and Column:** FIN_FINANCIALS.FORECAST_REV_FINISH  
**Type:** Date

plannedNPV
Optional. If the Calculate Present Value Info flag is set, this is recalculated after import.
**Table and Column:** FIN_FINANCIALS.PLANNED_NPV  
**Type:** Double

plannedROI
Optional. If the Calculate Present Value Info flag is set, this is recalculated after import. 1.0 is 100%.
**Table and Column:** FIN_FINANCIALS.PLANNED_ROI  
**Type:** Double
plannedBreakEven
Optional. This is recalculated after import if Calculate Present Value Info flag is set.

**Table and Column:** FIN_FINANCIALS.PLANNED_BREAKEVEN  
**Type:** Date

budgetNPV
Optional. If the Calculate Present Value Info flag is set, this is recalculated after import.

**Table and Column:** FIN_FINANCIALS.BUDGET_NPV  
**Type:** Double

budgetROI
Optional. If the Calculate Present Value Info flag is set, this is recalculated after import. 1.0 is 100%.

**Table and Column:** FIN_FINANCIALS.BUDGET_ROI  
**Type:** Double

budgetBreakEven
Optional. This is recalculated after import if Calculate Present Value Info flag is set.

**Table and Column:** FIN_FINANCIALS.BUDGET_BREAKEVEN  
**Type:** Date

forecastNPV
Optional. If the Calculate Present Value Info flag is set, this is recalculated after import.

**Table and Column:** FIN_FINANCIALS.FORECAST_NPV  
**Type:** Double

forecastROI
Optional. If the Calculate Present Value Info flag is set, this is recalculated after import. 1.0 is 100%.

**Table and Column:** FIN_FINANCIALS.FORECAST_ROI  
**Type:** Double

forecastBreakEven
Optional. This is recalculated after import if Calculate Present Value Info flag is set.

**Table and Column:** FIN_FINANCIALS.FORECAST_BREAKEVEN  
**Type:** Date
Forecast equals budget

Optional. If selected, forecast values will track the budget values.

**Table and Column:** FIN_FINANCIALS.IS_FCST_EQ_BDGT

**Type:** Boolean

Currency ISO code

Optional. A three-character ISO currency code that must be an active currency in CA Clarity.

**Table and Column:** FIN_FINANCIALS.CURRENCY_CODE

**Type:** String

Calculate present value info

Optional. If checked, this calculates the NPV, ROI, and breakeven after import.

**Table and Column:** FIN_FINANCIALS.IS_CALC_PV_INFO

**Type:** Boolean

**Service Schema Tag**

The service tag is part of the schema mapping for the Non-project Investment XOG object. It has the following attribute:

**Application Code**

Optional. Defines the unique identifier for the service.

**Table and Column:** INV_SERVICES.APPLICATION_ID

**Type:** String

**Schema Name**

nikuxog_service.xsd

**Read and Write XML Files**

The following XML files are included:

- inv_services_read.xml. Use this file to export service instances from CA Clarity.
- inv_services_write.xml. Use this file to import service instances that were previously exported from CA Clarity.
Notifications

Use the Notification XOG object to view inbound and outbound configurable notification templates.

Schema Name

nikuxog_notification.xsd

Read and Write XML Files

The following XML files are included:

- notification_read.xml. Use this file to export notification templates from CA Clarity.
- notification_write.xml. Use this file to import notification templates that were previously exported from CA Clarity.

Prerequisites

None.

Read Filters

The following explicit read filters are used:

functionalAreaCode

Defines the functional area code that refers to the notification template. The values can be obtained from the CMN_LOOKUPS_V table with lookup_type set to ‘NOTIFICATION_TYPE’.

notificationCode

Defines the ID of the notification.

languageCode

Defines the language code of the notification for extracting the name and description fields. This parameter is not required if reading the notification in all languages.

Error Handling

No errors are thrown. If the filter includes invalid values, no records are read.

Schema Mapping

Mappings for the following schema tag names are provided:

- Notification
Notification Schema Tag

The notification tag is part of the schema mapping for the notification XOG object. It has the following attributes:

notificationCode

Required. Defines the code for the notification template.

Table and Column: CMN_NOTIFICATIONS.NOTIFICATION_CODE

Type: String

functionalAreaCode

Required. Defines the functional area code that refers to the notification template.

Table and Column: CMN_NOTIFICATIONS.FUNCTIONAL_AREA_ID

Type: String

isCustomized

Identifies if the notification template is customized or not.

Table and Column: CMN_NOTIFICATIONS.IS_CUSTOMIZED

Type: Boolean

notificationNameDesc

Required. Defines the name and description of the notification template.

Table and Column: CMN_CAPTIONS_NLS.NAME and DESCRIPTIONS with TABLE_NAME mapped to CMN_NOTIFICATIONS

Type: String

notificationSubjectBody

Required. Defines the subject and message body text of the notification template.

Table and Column: CMN_CAPTIONS_NLS.NAME and DESCRIPTIONS with TABLE_NAME mapped to CMN_NOTIFICATIONS.

Type: String
**Portfolio**

Use the portfolio XOG object for outbound (read) and inbound (write) processing.

**Schema Name**

nikuxog_pfm_portfolio.xsd

**Read and Write XML Files**

The following files are included.
- **portfolio_read.xml**. Use this file to export portfolios data.
- **portfolio_write.xml**. Use this file to import portfolio data previously exported from CA Clarity.

**Business Rules and Processing**

The Portfolios schema is defined for inbound (write) processing only. Some of the portfolio fields (for example, budgetType, capacityType, capacityUnitType, invType) are optional on the schema but are required fields in the database table. If they are not provided in the input xml, the default values are used for these fields.

**Schema Mappings**

The following schema mappings are described:
- **Portfolios** (see page 315)
- **Contents** (see page 319)

**Portfolio Schema Tag**

The portfolio tag is part of the schema mapping for the portfolio XOG object. It has the following attributes:

- **code**
  - Required. Defines the unique ID of this portfolio.
  - **Table and Column**: pfm_portfolio.code
  - **Type**: String

- **name**
  - Required. Defines the name of this portfolio.
  - **Table and Column**: pfm_portfolio.name
  - **Type**: String
description
Optional. Defines the description of this portfolio.
*Table and Column:* `pfm_portfolio.description`
*Type:* String

*syncschedule_cron*
Optional. Defines the portfolio synchronization cadance in a cron like format.
*Table and Column:* `pfm_portfolio.syncschedule_cron`
*Type:* String

department
Optional. Defines the department (if any) associated with the portfolio.
*Table and Column:* `pfm_portfolio.department_id`
*Type:* String

currency_code
Optional. Defines the currency code for the portfolio.
*Table and Column:* `pfm_portfolio.currency_code`
*Type:* String

is_active
Optional. Defines the active status of the portfolio.
*Table and Column:* `pfm_portfolio.is_active`
*Type:* Boolean

*page_layout*
Optional. Defines the page layout of the portfolio.
*Table and Column:*
*Type:* String

*partition_code*
Optional. Defines the partition where the portfolio resides.
*Table and Column:* `odf_ca_pfm_portfolio.partition_code`
*Type:* String

cap_unit_type
Optional. Defines the capacity unit type for the portfolio (FTE or Hours).
*Table and Column:* `pfm_portfolio.capacity_unit_type`
*Type:* String
operating_cost_target
Optional. Defines the operating cost target for the portfolio.
Table and Column: pfm_portfolio.operating_cost_target
Type: Float

benefits_target
Optional. Defines the benefits target for the portfolio.
Table and Column: pfm_portfolio.benefits_target
Type: Float

resources_target
Optional. Defines the roles target for the portfolio.
Table and Column: pfm_portfolio.resources_target
Type: Float

operating_cost_curve
Optional. Defines the operating cost target for the portfolio as a curve.
Table and Column: pfm_portfolio.operating_cost_curve
Type: CurveType

capital_cost_curve
Optional. Defines the capital cost target for the portfolio as a curve.
Table and Column: pfm_portfolio.capital_cost_curve
Type: CurveType

benefits_curve
Optional. Defines the benefits target for the portfolio as a curve.
Table and Column: pfm_portfolio.benefits_curve
Type: CurveType

resources_curve
Optional. Defines the roles target for the portfolio as a curve.
Table and Column: pfm_portfolio.resources_curve
Type: CurveType

manager
Required. Defines the User ID of the portfolio manager who will manage this portfolio.
Table and Column: PMA_PORTFOLIOS. MANAGER_ID
Type: String
id
   Required. Defines the unique identifier for this portfolio.
   **Table and Column:** PMA_PORTFOLIOS.ID
   **Type:** String

parentPortfolioID
   Optional. Defines the unique identifier of the parent portfolio (if this a child
   portfolio).
   **Table and Column:** PMA_PORTFOLIOS.PARENT_ID
   **Type:** String

budgetCost
   Required. Defines the budgeted cost of this portfolio.
   **Table and Column:** PMA_PORTFOLIO.BDGT_CST_TOTAL
   **Type:** Number

budgetBenefit
   Required. Defines the budgeted benefit of this portfolio.
   **Table and Column:** PMA_PORTFOLIOS.BDGT_REV_TOTAL
   **Type:** Number

budgetType
   Optional. Defines the budget type for this portfolio.
   **Values:**
   - BUDGET_TYPE_TOTAL
   - BUDGET_TYPE_REMAINING
   **Default:** BUDGET_TYPE_TOTAL
   **Table and Column:** PMA_PORTFOLIOS.BUDGET_TYPE
   **Type:** String

capacityType
   Optional. Defines the capacity type for this portfolio.
   **Values:**
   - CAPACITY_TYPE_TOTAL
   - CAPACITY_TYPE_REMAINING
   **Default:** CAPACITY_TYPE_TOTAL
   **Table and Column:** PMA_PORTFOLIOS.CAPACITY_TYPE
   **Type:** String
invType

Optional. Defines the investment type for this portfolio.

Values: all, application, asset, idea, other, product, project, and service

Default: all

Table and Column: PMA_PORTFOLIOS. PORTFOLIO_INV_TYPE

Type: String

start_date

Required. Defines the start date for the portfolio horizon.

Table and Column: pfm_portfolio.start_date

Type: dateTime

finish_date

Required. Defines the finish date for the portfolio horizon.

Table and Column: pfm_portfolio.finish_date

Type: dateTime

currencyISOcode

Required.

Table and Column: PMA_PORTFOLIOS.CURRENCY_CODE

Type: String

pageLayoutCode

Optional.

Table and Column: CMN_INSTANCE_PAGES. PAGE_FRAME_ID

Type: String

Contents Schema Tag

This tag is part of the schema mapping for the Portfolio XOG object. It has the following attributes:

isIncluded

Defines whether portfolio content types are included.

Table and Column: PMA_PRTFLO_INCL_CTNT_TYPES.IS_INCLUDED

investmentType

Required. On include tag.

Table and Column: PMA_PRTFLO_INCL_CTNT_TYPES.OBJECT_TYPE_CODE
**lastSyncDate**

Defines the date the portfolio contents were last updated.

*Table and Column:* PMA_PRTFLIO_INCL_CTNT_TYPES.LAST_SYNC_DATE

**runSync**

This value is used only while processing.

*Table and Column:* Not Applicable

**investmentID**

Required. On contents/include/expression/investment tag and contents/investment tag.

*Table and Column:* PMA_PORTFOLIO_CONTENTS.INVEST_ID

**investmentType**

Required. On contents/include/expression/investment tag and contents/investment tag.

*Table and Column:* PMA_PORTFOLIO_CONTENTS.INVEST_TYPE
Process Notification

Use the Process Notification XOG object to view inbound and outbound configurable notification templates for processes.

Schema Name

nikuxog_processnotification.xsd

Read and Write XML Files

The following XML files are included:
- processnotification_read.xml. Use this file to export notification templates from <CA PPM>.
- processnotification_write.xml. Use this file to import notification templates that were previously exported from CA Clarity.

Prerequisites

Before using this XOG object, make sure the referenced object, such as the process exists in CA Clarity.

Read Filters

The following explicit read filters are used:

notificationCode
  Defines the ID of the notification template.

languageCode
  Defines the language code of the notification template for extracting the name and description fields. The parameter is not required if reading the notification template in all languages.

processCode
  Defines the process code for which the notification template was defined.

Error Handling

No errors are thrown. If the filter includes invalid values, no records are read.

Schema Mapping

Mappings for the following schema tag names are provided:
- Process Notification
Process Notification Schema Tag

The process notification tag is part of the schema mapping for the process notification XOG object. It has the following attributes:

**notificationCode**
- Required. Defines the code for the notification template.
- **Table and Column:** CMN_PROCESS_NOTIFICATIONS.NOTIFICATION_CODE
- **Type:** String

**functionalAreaCode**
- Required. Defines the functional area code that refers to the notification template.
- **Table and Column:** CMN_PROCESS_NOTIFICATIONS.FUNCTIONAL_AREA_ID to the CMN_LOOKUPS_V table with lookup_type set to NOTIFICATION_TYPE
- **Type:** String

**isCustomized**
- Defines if the notification template is customized or not.
- **Table and Column:** CMN_PROCESS_NOTIFICATIONS.IS_CUSTOMIZED
- **Type:** Boolean

**notificationNameDesc**
- Required. Defines the name and description of the notification template.
- **Table and Column:** CMN_CAPTIONS_NLS.NAME and DESCRIPTIONS with TABLE_NAME mapped to CMN_PROCESS_NOTIFICATIONS.
- **Type:** String

**notificationSubjectBody**
- Required. Defines the subject and message body texts of the notification template.
- **Table and Column:** CMN_CAPTIONS_NLS.NAME and DESCRIPTIONS with TABLE_NAME mapped to CMN_PROCESS_NOTIFICATIONS.
- **Type:** String
process_code

Required. Defines the process code to which the current notification is associated with.

**Table and Column:** BPM_DEF_PROCESSES.PROCESS_CODE that associates to the CMN_PROCESS_NOTIFICATIONS.PROCESS_ID

**Type:** String
Project

Use the Project XOG object to view inbound and outbound project attributes. Projects are defined for inbound (write) and outbound (read) processing. Common reasons to import and export project data include:

- To deliver project plans to drive external systems such as time sheet tools.
- To integrate with Oracle Projects and other project systems.
- To supply data for sophisticated reporting requirements.

Schema Name

nikuxog_project.xsd

Read and Write XML Files

The following XML files are included:

- prj_projects_read.xml. Use this file to export project attributes from CA Clarity.
- prj_projects_write.xml. Use this file to import business processes that were previously exported from CA Clarity.

Import Rules

When importing project management information, the user importing the project (XOG user) or the user passed in the Addedby attribute on the project XML, must have the Project - Edit Management - All access right.

The following import rules are applied while importing or updating project financial information:

- The Username attribute must be passed and the user must be financially-enabled.
- If the project is financially approved, changes to Client, Type, Billing Currency, Forecasting flag Template Flag, Billing Project Code and Exchange Rate Type are not allowed.
- If there are transactions on PPA_WIP (with amounts not equal to zero) for a project, then the changing of Status to Close for that project is not allowed.
- The Client, Project Class, WIP Class, Location, Department, and Currency used for a project must exist.
- Contract Type, Project Document Number, Contract Amount, and Date are required.
- Bill Amount, Number of Bills, Bill Frequency, and First Bill Date are entered once and you cannot change the value once imported.
- If the rate/cost matrix is used in project financials, the specified rate matrix must be validated along with all the columns mentioned in the rate or cost matrix.
If any transactions exist on PPA_TRANSCONTROL, then the status of the report cannot be changed to "closed".

**Deletion Rules**

You can delete tasks, assignments, and resources by passing the Deletion attribute (which takes True or False). Only the following delete operations are allowed:

- **Delete Task**: This also deletes all assignments for the task.
- **Remove a staff member from a project**: When you remove staff from a project team, it also removes assignments for the resource.
- **Remove assignment from task**: The deletes the assignment from the task.

You cannot delete teams, tasks, or assignments that are referenced by a time entry with non-zero actuals on a submitted timesheet (with a status other than Not Submitted or Rejected).

**Exporting Project Data**

When exporting project data from CA Clarity, the following elements are included:

- Basic project information
- Tasks
- Assignments
- Management information
- Financial information
Read Filters

The following arguments are taken:

- Tasks: include_tasks
- Resources: include_resources
- Baselines: include_baselines
- Allocations: include_allocations
- Custom: include_custom
- Tasks: include_tasks
- Estimate date: include_estimate_after_date
- Estimates: include_estimates
- Actuals: include_actuals
- Dependencies: include_dependencies
- Subprojects: include_subprojects

Note the following when using these arguments:

- When include_tasks and include_resources are Off, only project master record and financial information is exported.
- When include_tasks and include_resources arguments are On, all project information is exported, including assignments.
- When include_tasks is On and include_resources is Off, the Work Breakdown Structure (WBS) is exported, but not assignments or resource information.
- When include_tasks is Off and include_resources is On, project-level resource assignments are exported, but not the WBS or task-level assignments.

The following implicit filter is used:

- The project is enabled for management.

The following explicit filters are used:

- projectID
- approved (Management)
- approvedForBilling (Financial)
- start date
- end date
- lastUpdatedDate: This is the most recent date of lastUpdatedDate in the SRM_PROJECTS table or PRModTime in the PRJ_PROJECTS table.
- active project
management functionality is enabled

Schema Mappings - Outbound

Schema mappings are described for the following outbound Project tag names:

- SRM_PROJECTS
- SRM_RESOURCES (Manage Resource ID)
- PRJ_PROJECTS
- PAC_MNT_PROJECTS
- Project Location
- CLNTSUPP
- PRTEAM
- SRM_RESOURCES (Team Resource ID)
- SRM_RESOURCES (Project Role ID)
- SRM_RESOURCES (Assignment Resource ID)
- Phttp://jakarta.apache.org/commonsRTask
- PRAssignment
- OBS Association
- OBSAssoc

Schema Mappings - Inbound

Schema mappings are described for the following inbound Project tag names:

- PRJ_PROJECTS
- PAC_MNT_PROJECTS
- Resource
- Task
- TaskLabor

SRM_PROJECTS Schema Tag

This tag is part of the outbound schema mapping for the project XOG object.

**projectID**

Required.

**Table and Column:** SRM_PROJECTS.UNIQUE_NAME

**Type:** String
**name**

Required.

*Table and Column:* SRM_PROJECTS.NAME

*Type:* String

**description**

Optional.

*Table and Column:* SRM_PROJECTS.DESCRIPTION

*Type:* String

**createdBy**

Optional.

*Table and Column:* SRM_PROJECTS.CREATED_BY

*Type:* String

**createdDate**

Optional.

*Table and Column:* SRM_PROJECTS.CREATED_DATE

*Type:* dateTime

**lastUpdatedDate**

Optional.

This uses PRJ_PROJECTS.prModTime

*Table and Column:* SRM_PROJECTS.LAST_UPDATED_DATE

*Type:* dateTime

**lastUpdatedBy**

Optional.

This uses PRJ_PROJECTS.prModBy if PRJ_PROJECTS.prModTime is later than LAST_UPDATED_BY.

*Table and Column:* SRM_PROJECTS.LAST_UPDATED_BY

*Type:* String

**Active**

Optional.

*Table and Column:* SRM_PROJECTS.IS_ACTIVE

*Type:* Boolean
program
Optional. Indicates if the project is a program

**Table and Column:** SRM_PROJEC.TS.IS_PROGRAM
**Type:** Boolean

**SRM_RESOURCES Schema Tag**

The SRM_RESOURCES tag is part of the outbound schema mapping for the project XOG object. This tag uses the SRM_RESOURCES table, where SRM_RESOURCES.USER_ID = CMN_SEC_USERS.ID and CMN_SEC_USERS.USER_NAME = PRJ_PROJ EC.TS.MANAGER_ID.

This schema tag has the following attribute:

manageResourceId
Optional

**Table and Column:** SRM_RESOURCES.UNIQUE_NAME
**Type:** String

**Project (PRJ_PROJECTS) Schema Tag**

The PRJ_PROJECTS tag is part of the outbound schema mapping for the project XOG object. It has the following attributes:

start
Optional.

**Table and Column:** PRJ_PROJECTS.prStart
**Type:** date-time

finish

**Table and Column:** PRJ_PROJECTS.prFinish
Optional.
**Type:** date-time

approved
Optional.

True only if prApproveTime is not null.

**Table and Column:** PRJ_PROJECTS.prApproveTime
**Type:** Boolean
OpenForTimeEntry
Optional.
Table and Column: PRJ_PROJECTS.prIsOpen
Type: Boolean

Closed
Optional.
True only if prClosedTime is not null.
Table and Column: PRJ_PROJECTS.prClosedTime
Type: Boolean

**Project (PAC_MNT_PROJECTS) Schema Tag**

The PAC_MNT_PROJECTS tag is part of the outbound schema mapping for the project XOG object. It has the following attributes:

financialLocation
Table and Column: PAC_MNT_PROJECTS.LOCATIONID
Type: String

financialDepartment
Optional.
Table and Column: PAC_MNT_PROJECTS.DEPARTCODE
Type: String

clientID
Optional.
Table and Column: PAC_MNT_PROJECTS.COMPANY_CODE
Type: String

affiliatedProjectID
Optional.
Table and Column: PAC_MNT_PROJECTS.AFFILIATEPROJECT
Type: String
financialStatus
  Optional.
  Table and Column: PAC_MNT_PROJECTS.STATUS
  Type: String

approvedForBilling
  Optional.
  Table and Column: PAC_MNT_PROJECTS.APPROVED
  Type: Boolean

billingType
  Defines the billing type.
  Values: S, I, and C
  Table and Column: PAC_MNT_PROJECTS.TYPE
  Type: String

billingCurrencyCode
  Optional.
  Table and Column: PAC_MNT_PROJECTS.BILLING_CURRENCY_CODE
  Type: String

costType
  Optional. Refers to Cost type for the investment. Possible values are OPERATING/CAPITAL. Default value is OPERATING.
  Table and Column: PAC_MNT_PROJECTS.cost_type
  Type: String

Project Location Schema Tag

The Project Location tag is part of the outbound schema mapping for the project XOG object. It uses the LOCATIONS table, where LOCATIONS.LOCATIONID = PAC_MNT_PROJECTS.LOCATIONID.

This schema tag has the following attribute.

Entity
  Optional
  Table and Column: LOCATIONS.ENTITY
  Type: String
Project (CLNTSUPP) Schema Tag

This tag is part of the outbound schema mapping for the Project XOG object.

This tag uses the CLNTSUPP table, where CLNTSUPP.COMPANY_CODE = PAC_MNT_PROJECTS.COMPANY_CODE.

It has the following attribute.

**clientName**
- Optional
- **Table and Column:** CLNTSUPP.COMPANY_NAME
- **Type:** String

Resource (PRTEAM) Schema Tag

This tag is part of the outbound schema mapping for the Project XOG object. It has the following attributes:

This tag uses the PRTeam table with zero to many records and where PRTeam.prProjectID = PRJ_PROJECTS.priID.

**availFrom**
- Optional
- **Table and Column:** PRTeam.prAvailStart
- **Type:** dateTime

**availTo**
- Optional
- **Table and Column:** PRTeam.prAvailFinish
- **Type:** dateTime

**openForTimeEntry**
- Optional
- **Table and Column:** PRTeam.prIsOpen
- **Type:** Boolean

**lastUpdatedBy**
- Optional
- **Table and Column:** PRTeam.prModBy
- **Type:** String
lastUpdatedDate
Optional
Table and Column: PRTeam.prModTime
Type: dateTime
capitalPercentage
Optional. Capitalization percent of planned cost at team level. The valid value of this attribute can be any float value from 0 (0%) to 1 (100%). The default value is 0.
Table and Column: PRTEAM.CAPITAL_PERCENTAGE
Type: Float

**SRM_RESOURCES Schema Tag**

The SRM_RESOURCES tag is part of the outbound schema mapping for the project XOG object. This tag uses the SRM_RESOURCES table where SRM_RESOURCES.ID = PRTeam.prResourceID.

This schema tag has the following attributes:
resourceID
Required.
Table and Column: SRM_RESOURCES.UNIQUE_NAME
Type: String

**SRM_RESOURCES Schema Tag**

The SRM_RESOURCES tag is part of the outbound schema mapping for the project XOG object. This tag uses the SRM_RESOURCES table, where SRM_RESOURCES.ID = PRTeam.prRoleID.

This schema tag has the following attribute:
projectRoleID
Optional.
Table and Column: SRM_RESOURCES.UNIQUE_NAME
Type: String
**SRM_RESOURCES Schema Tag**

The SRM_RESOURCES tag is part of the outbound schema mapping for the project XOG object. This tag uses the SRM_RESOURCES table where SRM_RESOURCES.ID = PRAssignment.prResourceID.

This tag has the following attribute:

**resourceID**
Required.

*Table and Column:* SRM_RESOURCES.UNIQUE_NAME  
*Type:* String

**Task (PRTask) Schema Tag**

This tag is part of the outbound schema mapping for the Project XOG object.

This tag uses the PRTask table, with 0 to many records, and where PRTask.prProjectID = PRJ_PROJECTS.prID.

The Task (PRTASK) schema tag has the following attributes:

**taskID**
Optional. Defines the unique identifier for the task.  
*Table and Column:* PRTask.prExternalID  
*Type:* String

**name**
Optional. Defines the name for the task.  
*Table and Column:* PRTask.prName  
*Type:* String

**orderID**
Optional. Defines the unique order identifier for the task.  
*Table and Column:* PRTask.prWBSSequence  
*Type:* Integer

**outlineLevel**
Optional. Defines the outline level for the task.  
*Table and Column:* PRTask.prWBSLevel  
*Type:* Integer
start
Optional. Defines the start date and time for the task.
Table and Column: PRTask.prStart
Type: dateTime

baseStart
Optional. Defines the base start date and time for the task.
Table and Column: PRTask.prBaseStart
Type: dateTime

finish
Optional. Defines the finish date and time for the task.
Table and Column: PRTask.prFinish
Type: dateTime

baseFinish
Optional. Defines the base finish date and time for the task.
Table and Column: PRTask.prBaseFinish
Type: dateTime

milestone
Optional.
Table and Column: PRTask.prIsMilestone
Type: Boolean

summary
Optional. True only if prIsTask is false.
Table and Column: Not applicable
Type: Boolean

key
Optional.
Table and Column: PRTask.prIsKey
Type: Boolean

category
Optional.
Table and Column: PRTask.prCategory
Type: String
status
Optional. Defines the status of the task.

Values:
- 0. Not Started
- 1. Started
- 2. Complete

Table and Column: PRTask.prStatus
Type: Integer

percComp
Optional. Valid values are between 0 and 1 inclusive, where (100% is shown as 1.0.

Table and Column: PRTask.prPctComplete
Type: Float

lastUpdatedBy
Optional.

Table and Column: PRTask.prModBy
Type: String

lastUpdatedDate
Optional.

Table and Column: PRTask.prModTime
Type: dateTime

costType
Optional. Refers to Cost type for the task. Possible values are OPERATING/CAPITAL.

Table and Column: PRTASK.cost_type
Type: String
**TaskLabor (PRAssignment) Schema Tag**

This tag is part of the outbound schema mapping for the Project XOG object. It has the following attributes:

This tag uses the PRAssignment table, which can have 0 to many records, and where PRAssignment.prTaskID = PRTask.priID.

**start**
- Optional.
  - **Table and Column:** PRAssignment.prStart
  - **Type:** dateTime

**finish**
- Optional.
  - **Table and Column:** PRAssignment.prFinish
  - **Type:** dateTime

**actualWork**
- Optional.
  - **Table and Column:** PRAssignment.prActSum
  - **Type:** Float

**remainingWork**
- Optional.
  - **Table and Column:** PRAssignment.prEstSum
  - **Type:** Float

**baselineWork**
- Optional.
  - **Table and Column:** PRAssignment.prBaseSum
  - **Type:** Float

**actualThrough**
- Optional.
  - **Table and Column:** PRAssignment.prActThru
  - **Type:** dateTime
lastUpdatedBy

Optional.

Table and Column: PRAssignment.prModBy

Type: String

lastUpdatedDate

Optional.

Table and Column: PRAssignment.prModTime

Type: dateTime

OBS Association Schema Tag

The OBS Association tag is part of the outbound schema mapping for the Project XOG object. This schema tag is a wrapper for the OBSAssoc elements. No tables are used.

This tag has the following attribute:

completed

Optional. When completed and this value is True. Existing OBS associations not listed in the import are deleted.

Default: False

Table and Column: Not applicable

Type: String

OBS Association (OBSAssoc) Schema Tag

The OBS association tag is part of the outbound schema mapping for the project XOG object. The following tables are used in this tag:

- PRJ_OBS_ASSOCIATIONS (OBS associations)
- PRJ_OBS_TYPES (OBS)
- PRJ_OBS_UNITS (OBS Units)
- PRJ_OBS_UNITS_FLAT (OBS Units Flat Hierarchy)

This tag has the following attributes:

id

Required.

Table and Column: PRJ_OBS_TYPES.UNIQUE_NAME

Type: String
name
Optional.
Table and Column: PRJ_OBS_TYPES.PRJ_OBS_TYPES.NAME
Type: String

unitPath
Required. This is a slash-delimited list of unit names that points to the unit to which the object is associated.
Table and Column: PRJ_OBS_TYPES.PRJ_OBS_UNITS.NAME
Type: String
Example: CAN/BC/VAN

Project (PRJ_PROJECTS) Schema Tag

The PRJ_PROJECTS tag is part of the inbound schema mapping for the project XOG object. It has the following attributes:

Name
Required.
Table and Column: PRJ_PROJECTS.NAME
Type: String

projectID
Required. Uniquely identifies this project and is used as key for sub-project references.
Table and Column: PRJ_PROJECTS.UNIQUE_NAME
Type: String

description
Optional.
Table and Column: PRJ_PROJECTS.DESCRIPTION
Type: String

AddedBy
Optional. When AddedBy (CreatedBy in CA Clarity), the default Collaboration Manager is assigned to the project.
Table and Column: PRJ_PROJECTS.CREATED_BY
Type: String
**AddedDate**

Optional.

**Table and Column:** PRJ_PROJECTS.CREATED_DATE  
**Type:** dateTime

**lastUpdatedDate**

Optional. Uses PRJ_PROJECTS.prModTime if later.

**Table and Column:** PRJ_PROJECTS.LAST_UPDATE_DATE  
**Type:** dateTime

**lastUpdatedBy**

Optional. Uses PRJ_PROJECTS.prModBy if PRJ_PROJECTS.prModTime is later than LAST_UPDATED_BY.

**Table and Column:** PRJ_PROJECTS.LAST_UPDATED_BY  
**Type:** String

**Active**

Optional.

**Table and Column:** PRJ_PROJECTS.IS_ACTIVE  
**Type:** Boolean

**Type**

Required.

**Table and Column:** PRJ_PROJECTS.TYPE  
**Type:** String

**stageIdentifier**

Required.

**Table and Column:** PRJ_PROJECTS.STAGE_IDENTIFIER  
**Type:** String

**start**

Required.

**Table and Column:** PRJ_PROJECTS.prStart  
**Type:** dateTime

**finish**

Required.

**Table and Column:** PRJ_PROJECTS.prFinish  
**Type:** dateTime
approved
Optional. True only if prApprovedTime is not null. On import, prApprovedTime is set to the current time if it was previously null.

**Table and Column:** None

**Type:** Boolean

openForTimeEntry
Optional.

**Table and Column:** PRJ_PROJECTS.prIsOpen

**Type:** Boolean

closed
Optional. True only if prClosedTime is not null. On import, prClosedTime is set to the current time if it was previously null.

**Table and Column:** None

**Type:** Boolean

guidelines
Optional.

**Table and Column:** PRJ_PROJECTS.prGuidelines

**Type:** String

department
Optional.

**Table and Column:** PRJ_PROJECTS.prDepartment

**Type:** String

asOf
Optional.

**Table and Column:** PRJ_PROJECTS.prAsOf

**Type:** dateTime

cpmType
Optional.

**Table and Column:** PRJ_PROJECTS.prCPMType

**Type:** Integer
trackMode
Optional.
**Table and Column:** PRJ_PROJECTS.prTrackMode
**Type:** Integer

sponsoredBy
Optional.
**Table and Column:** PRJ_PROJECTS.prSponsoredBy
**Type:** String

requestedBy
Optional.
**Table and Column:** PRJ_PROJECTS.prRequestedBy
**Type:** String

requestedTime
Optional.
**Table and Column:** PRJ_PROJECTS.prRequestedTime
**Type:** dateTime

userText1
Optional.
**Table and Column:** PRJ_PROJECTS.prUserText1
**Type:** String

userText2
Optional.
**Table and Column:** PRJ_PROJECTS.prUserText2
**Type:** String

userText3
Optional.
**Table and Column:** PRJ_PROJECTS.prUserText3
**Type:** String

userText4
Optional.
**Table and Column:** PRJ_PROJECTS.prUserText4
**Type:** String
userText5
Optional.
Table and Column: PRJ_PROJECTS.prUserText5
Type: String

userText6
Optional.
Table and Column: PRJ_PROJECTS.prUserText6
Type: String

userText7
Optional.
Table and Column: PRJ_PROJECTS.prUserText7
Type: String

Format
Optional.
Table and Column: PRJ_PROJECTS.prFormat
Type: Integer

Priority
Optional. Defines the priority.
Values: 0-36, where:
■ 0. Highest priority
■ 36. Lowest priority
Table and Column: PRJ_PROJECTS.prPriority
Type: Integer

username
Optional. On import, the ProjectManagerMgmt access right is granted to this user.
Table and Column: PRJ_PROJECTS.prUsername
Type: String

startImposed
Optional.
Table and Column: PRJ_PROJECTS.prStartImposed
Type: Boolean
**finishImposed**
Optional.
*Table and Column:* PRJ_PROJECTS.prFinishImposed
*Type:* Boolean

**baseTime**
Optional.
*Table and Column:* PRJ_PROJECTS.prBaseTime
*Type:* dateTime

**baseStart**
Optional.
*Table and Column:* PRJ_PROJECTS.prBaseStart
*Type:* dateTime

**baseFinish**
Optional.
*Table and Column:* PRJ_PROJECTS.prBaseFinish
*Type:* dateTime

**Chargecode**
Optional. Validate existence with PRChargeCode.prID
*Table and Column:* PRJ_PROJECTS.PrChargeCodeID
*Type:* String

**ManagerResource**
Optional. Must be a valid resource.
*Table and Column:* PRJ_PROJECTS.PrUserName
*Type:* String

---

**PAC_MNT_PROJECTS Schema Tag**

The PAC_MNT_PROJECTS tag is part of the inbound schema mapping for the project XOG object. It has the following attributes:

**financialLocation**
Defines the financial location for the project.
*Table and Column:* PAC_MNT_PROJECTS.LOCATIONID
*Type:* String
financialDepartment
Optional. Defines the financial location for the project.
Table and Column: PAC_MNT_PROJECTS.DEPARTCODE
Type: String

clientID
Defines the unique client identifier for the project. If other financial properties fields are present, then clientID is required. Otherwise, clientID and all financial properties are optional.
Table and Column: PAC_MNT_PROJECTS.COMPANY_CODE
Type: String

affiliatedProjectID
Optional. Defines the affiliated project unique identifier for the project.
Table and Column: PAC_MNT_PROJECTS.AFFILIATEPROJECT
Type: String

financialStatus
Optional. Defines the financial status for the project.
Table and Column: PAC_MNT_PROJECTS.STATUS
Type: String

approvedForBilling
Optional.
Table and Column: PAC_MNT_PROJECTS.APPROVED
Type: Boolean

billingType
Optional. Defines the billing type for the project.
Values: S, I.
Table and Column: PAC_MNT_PROJECTS.TYPE_
Type: String

billingCurrencyCode
Required only when CA Clarity is set for multi-currency. Defines the billing currency code for the project.
Table and Column: PAC_MNT_PROJECTS.BILLING_CURRENCY_CODE
Type: String
ProjectClass
Optional. Defines the project class for the project. This column defaults to clientID if
Projectclass is not supplied.

Table and Column: PAC_MNT_PROJECTS.CLASS
Type: String

WIP Class
Optional. Defines the WIP class for the project. This column defaults to clientID if
WIP Class is not supplied.

Table and Column: PAC_MNT_PROJECTS.WIPCLASS
Type: String

UseBudget
Optional.

Table and Column: PAC_MNT_PROJECTS.BUDGET
Type: Number

Billing Project
Optional.

Table and Column: PAC_MNT_PROJECTS.BILLING_PROJECT_ID
Type: Number

Send Bill To
Optional.

Table and Column: PAC_MNT_PROJECTS.BILL_TO_COMPANY_CODE
Type: String

billExpenses
Optional.

Table and Column: PAC_MNT_PROJECTS.EX_BILL_EXPENSES
Type: Number

Document number
Optional.

Table and Column: PAC_MNT_PROJECTS.CONTRACTNBR
Type: String
Contract amount
Optional.
Table and Column: PAC_MNT_PROJECTS.Contract Amount
Type: Number

Enforce Contract amount
Optional.
Values:
- 0. Do not enforce
- 1. Enforce
Default: 0
Table and Column: PAC_MNT_PROJECTS.ENFORCE_CONTRACT_AMOUNT
Type: Number

Date
Optional.
Table and Column: PAC_MNT_PROJECTS.Contract Date
Type: Date

Labor Rate Source
Optional.
Table and Column: PAC_MNT_PROJECTS.TRANSRATESOURCELABOR
Type: Number

Labor Cost Source
Optional.
Table and Column: PAC_MNT_PROJECTS.TRANSCOSTSOURCELABOR
Type: Number

Labor Exchange Rate
Optional.
Table and Column: PAC_MNT_PROJECTS.LABOR_EXCHANGE_RATE_TYPE
Type: Number

Materials Rate Source
Optional.
Table and Column: PAC_MNT_PROJECTS.TRANSRATESOURCENEW
Type: Number
Materials Cost Source
Optional.
*Table and Column:* PAC_MNT_PROJECTS.TRANSCOSTSOURCEMATERIALS
*Type:* Number

Materials Exchange Rate
Optional.
*Table and Column:* PAC_MNT_PROJECTS.MATERIALS_EXCHANGE_RATE_TYPE
*Type:* Number

EXPENSES Rate Source
Optional.
*Table and Column:* PAC_MNT_PROJECTS.TRANSRATESOURCELABOR
*Type:* Number

EXPENSES Cost Source
Optional.
*Table and Column:* PAC_MNT_PROJECTS.TRANSCOSTSOURCE EXPENSES
*Type:* Number

EXPENSES Exchange Rate
Optional.
*Table and Column:* PAC_MNT_PROJECTS.EXPENSE_EXCHANGE_RATE_TYPE
*Type:* Number

EQUIPMENT Rate Source
Optional.
*Table and Column:* PAC_MNT_PROJECTS.TRANSRATESOURCE EQUIPMENT
*Type:* Number

EQUIPMENT Cost Source
Optional.
*Table and Column:* PAC_MNT_PROJECTS.TRANSCOSTSOURCE EQUIPMENT
*Type:* Number

EQUIPMENT Exchange Rate
Optional.
*Table and Column:* PAC_MNT_PROJECTS.EQUIPMENT_EXCHANGE_RATE_TYPE
*Type:* Number
Submitted for Approval

Optional. Set to 1 if the project financial properties is to be marked submitted for approval.

**Table and Column:** PAC_MNT_PROJECTS.AWAITINGAPPROVAL

**Type:** Number

Submitted and Approved

Optional. Set to 1 if the project financial properties is to be marked submitted and approved.

**Table and Column:** PAC_MNT_PROJECTS.APPROVED

**Type:** Number

batchCycle

Optional. Defines the batch cycle associated with the project.

**Table and Column:**

**Type:** String

**Resource (PRTeam) Schema Tag**

This tag is part of the inbound schema mapping for the Project XOG object. PRTeam is populated when you staff a resource to a project.

This tag uses the PRTeam table, which can have 0 to many records and where PRTeam.prProjectID = PRJ_PROJECTS.prID.

The Resource (PRTeam) schema tag has the following attributes:

**availFrom**

Optional. Uses the PRJ_PROJECTS.prStart if prAvailStart is not set.

**Table and Column:** PRTeam.prAvailStart

**Type:** dateTime

**availTo**

Optional. Uses PRJ_PROJECTS.prFinish if prAvailFinish is not set.

**Table and Column:** PRTeam.prAvailFinish

**Type:** dateTime
openForTimeEntry
  Optional.
  Table and Column: PRTeam.prIsOpen
  Type: Boolean

lastUpdatedBy
  Optional.
  Table and Column: PRTeam.prModBy
  Type: String

lastUpdatedDate
  Optional.
  Table and Column: PRTeam.prModTime
  Type: dateTime

bookingStatus
  Optional.
  Table and Column: PRTeam.prBooking
  Type: Integer

requestStatus
  Optional.
  Table and Column: PRTeam.prStatus
  Type: Integer

defaultAllocation
  Optional. The default allocation for this team member.
  Table and Column: PRTeam.prAllocCurve
  Type: Float

resourceID
  Optional. May be more than one resource in the prTeam.
  Table and Column: PRTeam.prResourceId
  Type: String

ProjectRoleID
  Optional. May be more than one role in prTeam.
  Table and Column: PRTeam.PrRoleId
  Type: String
**PRTask Schema Tag**

This tag is part of the inbound schema mapping for the Project XOG object. PRTeam is populated when you staff a resource to a project.

This tag uses the PRTask table, which can have 0 to many records and where PRTask.prProjectID = PRJ_PROJECTS.prID.

The Task (PRTask) schema tag has the following attributes:

- **taskId**
  
  Optional. Defines the unique identifier for the task. Required, if referenced by Relation tag or as a sub-project.
  
  **Table and Column:** PRTask.prExternalID
  
  **Type:** String

- **name**
  
  Optional. Defines the name for the task.
  
  **Table and Column:** PRTask.prName
  
  **Type:** String

- **orderID**
  
  Optional.
  
  **Table and Column:** PRTask.prWBSSequence
  
  **Type:** Integer

- **outlineLevel**
  
  Optional.
  
  **Table and Column:** PRTask.prWBSLevel
  
  **Type:** Integer

- **start**
  
  Optional.
  
  **Table and Column:** PRTask.prStart
  
  **Type:** dateTime

- **baseStart**
  
  Optional.
  
  **Table and Column:** PRTask.prBaseStart
  
  **Type:** dateTime
finish
Optional.
**Table and Column:** PRTask.prFinish  
**Type:** dateTime

baseFinish
Optional.
**Table and Column:** PRTask.prBaseFinish  
**Type:** dateTime

milestone
Optional.
**Table and Column:** PRTask.prIsMilestone  
**Type:** Boolean

summary
Optional. This is set to True only if prIsTask is False.
**Table and Column:** Not Applicable  
**Type:** Boolean

key
Optional.
**Table and Column:** PRTask.prIsKey  
**Type:** Boolean

category
Optional.
**Table and Column:** PRTask.prCategory  
**Type:** String

status
Optional. Defines the status of the task.
**Values:**
- 0. Not Started
- 1. Started
- 2. Complete
**Table and Column:** PRTask.prStatus  
**Type:** Integer
percComp
Optional. Valid values are between 0 and 1 inclusive, where 100% is shown as 1.0.
Table and Column: PRTask.prPctComplete
Type: Float

lastUpdatedBy
Optional.
Table and Column: PRTask.prModBy
Type: String

lastUpdatedDate
Optional.
Table and Column: PRTask.prModTime
Type: dateTime

earlyStart
Optional.
Table and Column: PRTask.prEarlyStart
Type: dateTime

lateStart
Optional.
Table and Column: PRTask.prLateStart
Type: dateTime

earlyFinish
Optional.
Table and Column: PRTask.prEarlyFinish
Type: dateTime

lateFinish
Optional.
Table and Column: PRTask.prLateFinish
Type: dateTime

Duration
Optional.
Table and Column: PRTask.prDuration
Type: Float
**baselineDuration**
Optional.
*Table and Column: PRTask.prBaseDuration*
*Type: Float*

**totalSlack**
Optional.
*Table and Column: PRTask.prTotalFloat*
*Type: Float*

**Unplanned**
Optional. If the task is being updated, set this to False".
*Table and Column: PRTask.prIsUnplanned*
*Type: Boolean*

**shortName**
Optional.
*Table and Column: PRTask.prShortname*
*Type: String*

**Guidelines**
Optional.
*Table and Column: PRTask.prGuidelines*
*Type: String*

**Fixed**
Optional.
*Table and Column: PRTask.prIsFixed*
*Type: Boolean*

**Locked**
Optional.
*Table and Column: PRTask.prIsLocked*
*Type: Boolean*

**baseFixed**
Optional.
*Table and Column: PRTask.prBaseIsFixed*
*Type: Boolean*
**TaskLabor (PRAssignment) Schema Tag**

This tag is part of the inbound schema mapping for the Project XOG object. This PRAssignment table is populated when a resource is assigned to a task. This feature is not supported in CA Clarity but can be accomplished with Open Workbench or the CA Clarity Microsoft Project interface.

This tag uses the PRAssignment table, which can have 0 to many records, and where PRAssignment.prTaskID = PRTask.priD.

This tag has the following attributes:

**Start**

- Optional.
- **Table and Column:** PrAssignment.prStart
- **Type:** dateTime

**finish**

- Optional.
- **Table and Column:** PrAssignment.prFinish
- **Type:** dateTime

**actualWork**

- Optional.
- **Table and Column:** PrAssignment.prActSum
- **Type:** Float

**remainingWork**

- Optional.
- **Table and Column:** PrAssignment.prEstSum
- **Type:** Float

**baseTime**

- Optional.
- **Table and Column:** PRTask.prBaseTime
- **Type:** dateTime
baselineWork
Optionally.
**Table and Column:** PrAssignment.prBaseSum
**Type:** Float

actualThrough
Optionally.
**Table and Column:** PrAssignment.prActThru
**Type:** dateTime

lastUpdatedBy
Optionally.
**Table and Column:** PrAssignment.prModBy
**Type:** string

lastUpdatedDate
Optionally.
**Table and Column:** PrAssignment.prModTime
**Type:** dateTime

Unplanned
Optionally. During update Unplanned is False.
**Table and Column:** PrAssignment.prIsUnplanned
**Type:** Boolean

estPattern
Optionally.
**Table and Column:** PrAssignment.prEstPattern
**Type:** Integer

basePattern
Optionally.
**Table and Column:** PrAssignment.prBasePattern
**Type:** Integer

estMax
Optionally.
**Table and Column:** PrAssignment.prEstMax
**Type:** Float
baseMax
Optional.
Table and Column: PrAssignment.prBaseMax
Type: Float

pendEstSum
Optional.
Table and Column: PrAssignment.prPendEstSum
Type: Float

pendActSum
Optional.
Table and Column: PrAssignment.prPendActSum
Type: Float

pendStart
Optional.
Table and Column: PrAssignment.prPendStart
Type: dateTime

pendFinish
Optional.
Table and Column: PrAssignment.prPendFinish
Type: dateTime

Status
Optional.
Table and Column: PrAssignment.prStatus
Type: Integer

curveType
Required.
Table and Column: PrAssignment.prExtension
Type: Integer

Start
Required.
Table and Column: PrAssignment.prExtension
Type: dateTime
Finish

Required.

Table and Column: PrAssignment.prExtension
Type: dateTime

Sum

Required.

Table and Column: PrAssignment.prExtension
Type: Float

resourceID

Required.

Table and Column: PrAssignment.prResourceId
Type: String
**Requirement**

Use the requirement XOG object to view inbound and outbound requirements.

The requirement read XOG reads all data from existing requirements and writes them to an xml format. The requirement write XOG writes new requirements or updates existing requirements.

**Schema Name**

nikuxog_requirement.xsd

**Read and Write XML Files**

The following XML files are included:

- requirements_read.xml. Use this file to export requirements from CA Clarity.
- requirements_write.xml. Use this file to import requirements that were previously exported from CA Clarity.

**Prerequisites**

None

**Business Rules and Processing**

The following business rules and processing apply to this XOG:

- CA Clarity users must have the appropriate access rights to read or write requirements via XOG.
- Referenced investments, releases, and users (e.g. manager) must exist in CA Clarity prior to xogging in the requirement or they will not be added.
- Requirements are defined for inbound (write) and outbound (read) processing.

**Read Filters**

The following read filters are used:

- objectID
- investmentID
- managerName
- requirementTitle
- requirementStatus (New, Need Additional Information, In Pipeline, Active Candidate, Assigned to Release, Approved, Implemented, Duplicate, Rejected, and Draft)
Error Handling

The following fields are written to the Success and Error files when referenced objects such as users, investments, and releases or lookup values cannot be found, and the XOG process generates a warning:

- created_by
- currency_code
- last_updated_by
- manager
- requested_by
- risk
- status
- theme
- type
- release_id
- project_id
- task_name
- investment_id

If dependencies or hierarchy links are included in the inbound XOG file, a warning is written to the Success and Error files if either one of the associated requirements does not exist. Since you can add the missing requirement later to the file, this warning does not imply that the association will never be made when xogging in many requirements. The warning's intention is to provide information to the administrator so that they can confirm that all associations have been made.

Schema Mappings

The requirement element can contain link and dependency elements which relate requirements to one another. The following tables are mapped to requirements:

- RQP_RELEASES (see page 361)
- RQP_REQUIREMENTS (see page 361)
- RQP_REQ_DEPENDENCIES (see page 364)
- PRTASK (see page 365)
- SRM_RESOURCES (see page 365)
- INV_INVESTMENTS (see page 366)
**RQPRELEASES Schema Tag**

The RQPRELEASES tag is part of the schema mapping for the requirement XOG object. It has the following attributes:

- **release_id**
  - **Table and Column:** RQPRELEASES.CODE
  - **Type:** String

**RQPREQUIREMENTS Schema Tag**

The RQPREQUIREMENTS tag is part of the schema mapping for the requirement XOG object. It has the following attributes:

- **title**
  - Required.
  - **Table and Column:** RQPREQUIREMENTS.TITLE
  - **Type:** String

- **object_id**
  - Required. Must be unique.
  - **Table and Column:** RQPREQUIREMENTS.CODE
  - **Type:** String

- **child**
  - Required.
  - **Table and Column:** RQPREQUIREMENTS.CODE
  - **Type:** String

- **parent**
  - Required.
  - **Table and Column:** RQPREQUIREMENTS.CODE
  - **Type:** String

- **requirement**
  - Required.
  - **Table and Column:** RQPREQUIREMENTS.CODE
  - **Type:** String
dependent

Required.

**Table and Column:** RQP_REQUIREMENTS.CODE

**Type:** String

**description**

Required. Must be unique.

**Table and Column:** RQP_REQUIREMENTS.DESCRIPTION

**Type:** String

**committed**

Required. Must be unique.

**Table and Column:** RQP_REQUIREMENTS.COMMITTED

**Type:** Boolean

**est_op_cost**

**Table and Column:** RQP_REQUIREMENTS.EST_OP_COST

**Type:** Non-negative number

**est_cap_cost**

**Table and Column:** RQP_REQUIREMENTS.EST_CAP_COST

**Type:** Non-negative number

**est_effort**

**Table and Column:** RQP_REQUIREMENTS.EST_EFFORT

**Type:** Non-negative number

**est_hlm_size**

**Table and Column:** RQP_REQUIREMENTS.EST_HLM_SIZE

**Type:** Integer

**est_size**

**Table and Column:** RQP_REQUIREMENTS.EST_SIZE

**Type:** Non-negative number
bgt_op_cost
  Table and Column: RQP_REQUIREMENTS.BGT_OP_COST
  Type: Non-negative number

bgt_cap_cost
  Table and Column: RQP_REQUIREMENTS.BGT_CAP_COST
  Type: Non-negative number

bgt_effort
  Table and Column: RQP_REQUIREMENTS.BGT_EFFORT
  Type: Non-negative number

bgt_size
  Table and Column: RQP_REQUIREMENTS.BGT_SIZE
  Type: Non-negative number

priority_1
  Table and Column: RQP_REQUIREMENTS.PRIORITY_1
  Type: Integer (allowed range 0-100)

priority_2
  Table and Column: RQP_REQUIREMENTS.PRIORITY_2
  Type: Integer (allowed range 0-100)

priority_3
  Table and Column: RQP_REQUIREMENTS.PRIORITY_3
  Type: Integer (allowed range 0-100)

priority_4
  Table and Column: RQP_REQUIREMENT.PRIORITY_4
  Type: Integer (allowed range 0-100)

feature_overview
  Table and Column: RQP_REQUIREMENTS FEATURE_OVERVIEW
  Type: String

feature_driver
  Table and Column: RQP_REQUIREMENTS.FEATURE_DRIVER
  Type: String
**feature_vision**
*Table and Column:* RQP_REQUIREMENTS.FEATURE_VISION
*Type:* String

**req_impl_plan**
*Table and Column:* RQP_REQUIREMENTS.REQ_IMPL_PLAN
*Type:* String

**req_impact_analysis**
*Table and Column:* RQP_REQUIREMENTS.REQ_IMPACT_ANALYSIS
*Type:* String

**req_comments**
*Table and Column:* RQP_REQUIREMENTS.REQ_COMMENTS
*Type:* String

**story_text_1**
*Table and Column:* RQP_REQUIREMENTS.STORY_TEXT_1
*Type:* String

**story_text_2**
*Table and Column:* RQP_REQUIREMENTS.STORY_TEXT_2
*Type:* String

**story_text_3**
*Table and Column:* RQP_REQUIREMENTS.STORY_TEXT_3
*Type:* String

**RQP_REQ_DEPENDENCIES Schema Tag**

The RQP_REQ_DEPENDENCIES tag is part of the schema mapping for the requirement XOG object. It has the following attributes:

**is_mutual**
*Table and Column:* RQP_REQ_DEPENDENCIES.IS_MUTUAL
*Type:* Boolean
**PRTASK Schema Tag**

The PRTASK tag is part of the schema mapping for the requirement XOG object. It has the following attributes:

- **task_name**
  - Table and Column: PRTASK.PRNAME
  - Type: String

**SRM_RESOURCES Schema Tag**

The SRM_RESOURCES tag is part of the schema mapping for the requirement XOG object. It has the following attributes:

- **requested_by**
  - Table and Column: SRM_RESOURCES.UNIQUE_NAME
  - Type: String

- **manager**
  - Table and Column: SRM_RESOURCES.UNIQUE_NAME
  - Type: String

- **approved_by**
  - Table and Column: SRM_RESOURCES.UNIQUE_NAME
  - Type: String

- **created_by**
  - Table and Column: SRM_RESOURCES.UNIQUE_NAME
  - Type: String

- **last_updated_by**
  - Table and Column: SRM_RESOURCES.UNIQUE_NAME
  - Type: String
INV_INVESTMENTS Schema Tag

The INV_INVESTMENTS tag is part of the schema mapping for the requirement XOG object. It has the following attributes:

**project_id**

Defines the unique identifier for the project.

**Table and Column:** INV_INVESTMENTS.CODE  
**Type:** String

**investment_id**

Defines the unique identifier for the investment.

**Table and Column:** INV_INVESTMENTS.CODE  
**Type:** String

Resource

Use the resource XOG object to view inbound (write) and outbound (read) resource attributes. This XOG object does not allow you to create a resource for a specific partition; it only supports the _ROOT partition code. You can, however, use the content pack XOG to add a resource as a member of a partition.

Schema Name

nikuxog_resource.xsd

Read and Write XML Files

The following XML files are included:

- **rsm_resources_read.xml.** Use this file to export resource attributes from CA Clarity.
- **rsm_resources_write.xml.** Use this file to import resource attributes that were previously exported from CA Clarity.

Business Rules and Processing

Prior to importing resources, you must set up the following items correctly:
Manager

Defines the resource’s manager user name. This is a browse field against the CMN_SEC_USERS table. If the manager does not exist, the XML schema posts the resource without the manager’s username and a warning is posted to the Success and Error file. If the username exists, the XML schema fills the field. When this XML schema is run for the first time, many records may be posted without values for the manager. Subsequently when the schema is run, user profiles are updated to include a value for the manager (provided the user was added during the first pass).

Company ID

Defines the unique identifier for the company associated with the resource. This is a browse field against the SRM_COMPANIES table. If the company does not exist, the schema posts the resource, leaves this field blank, and posts a warning to the Success and Error file. If the company exists, the schema fills the field. When this schema is run the first time, many records are posted without values for the company. Subsequently when the schema is run, user profiles are updated to include the company information (provided the user was added during the first pass).

Financial Properties

The financial properties associated with the resource.

Before resources can be imported into CA Clarity, the following financial properties are set up in the Financial Administration module. If the financial properties are not found in CA Clarity, the resource’s financial properties are not imported into CA Clarity and an error is written to the Success and Error file. These properties are optional:

- Financial Location
- Financial Department
- Resource Class

Vendor

(This is an exception and is not required for the financial properties. The resource is added without a vendor and a warning is posted to the Success and Error file.)

Transaction Class

The transaction class associated with the resource. The value in the XOG can found in the TRANSCLASS table.

Management Properties

Defines the management properties associated with the resource. The XOG assumes a resource is associated with the standard calendar available for a typical work day's standard number of hours (based upon the standard calendar settings). The value is based on the default Allocation percentage.
OBS association

The OBS associated with the resource. A Security OBS is required for labor resources; all resources can have OBS associations. To accommodate this, there is an OBS Associations portlet that can be used for import and export.

Custom Attributes

The custom fields associated with the resource. The XOG allows for an unlimited number of custom-defined fields, however you must map the generated field to the schema. Within the schema for custom-defined fields, provide the Column Name, Attribute Name, and Value (since these can be modified by CA Clarity users).

Lookup values

Any lookup values associated with the resource. Lookup codes must be provided when appropriate. They are validated against CMN_LOOKUPS. Lookups are extracted for a set of lookup codes.

Primary Roleid

The primary role of this resource. This is a browse field against SRM_RESOURCES. If the primary Roleid passed does not exist, this field is left blank and a warning message is posted to the Success and Error file (the field is filled when the role exists).
Read Filters

The XOG allows for outbound processing of users based on the following fields:

- Resource Type
- Active
- And or Or processing is supported between the two fields and for processing within Type. The following combinations are supported:

  isActive = x
  
  where x = Active, Inactive

  ResourceType = x
  
  where x = LABOR, MATERIAL, EQUIPMENT or EXPENSE

  isActive = x AND ResourceType = y
  
  where x = Active, Inactive
  
  where y = one of many Resource Types

The following arguments are accepted:

- Contact: include_contact
- Management: include_management
- Financials: include_financial
- Custom Information: include_custom

The following statements concern these two arguments:

When all arguments are "Off", only resource basic information is exported.

When all arguments are "On", all resource information is exported.

Error Handling

The following fields are written to the Success and Error file when the XOG process generates an error or warning:

- externalId
- externalSource
- resourceld
- lastName

If an error occurs, the table is not updated. You must fix the error and run the XOG again. When a warning occurs because of inconsistencies in the data, the record is posted and the non-required fields are defaulted.

The following errors should be validated against the resource:
<table>
<thead>
<tr>
<th>Error or Warning Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourceId</td>
<td>The unique identifier for the resource. The resource is validated against the resourceId field. If the resource ID: ■ Is not unique, then the resource is not imported and an error is posted to the Success and Error file. ■ Unique, then the resource is imported.</td>
</tr>
<tr>
<td>managerUserName</td>
<td>The unique identifier for the manager associated with the resource. The manager is validated against the companyId field. If the manager: ■ Is not found, then the resource is imported without any association to a manager. ■ Exists, then it is also imported.</td>
</tr>
<tr>
<td>companyId</td>
<td>The unique identifier for the company associated with the resource. The company ID is validated against the companyId field. If the company: ■ Is not found, then the resource is imported without any association to a company. ■ Exists, then it is also imported.</td>
</tr>
<tr>
<td>vendorCode</td>
<td>The unique code for the vendor associated with the resource. The vendor code is validated against the vendorCode field. If the vendor: ■ Is not found, then the resource is imported without any association to a vendor. ■ Exists, then it is also imported.</td>
</tr>
<tr>
<td>PAC_MNT_RESOURCES</td>
<td>The financial properties related to the resource. If the financial properties: ■ Do not exist, then the resource is imported without any association to financial properties. ■ Exist, then they are also imported.</td>
</tr>
</tbody>
</table>
Schema Mappings

The following schema tags are described:

- Personal Information
- Contact Information
- Management Information
- Financial Information
- Expenses
- Rates and Costs
- Custom Information
- OBS Associations
- SkillAssocs

Personal Information Schema Tag

The personal information tag is part schema mapping for the resource XOG object. It has the following attributes:

ExternalSource

Required. Required by the schema lookup value. This is the originating system ID (for example Oracle)

Table and Column: SRM_RESOURCES.External_Source_ID
Type: String in the XML Schema) and Number in CA Clarity.

ExternalId

Required. Required by the XML schema. This is the originating unique identifier.

Table and Column: SRM_RESOURCES.External_ID
Type: String

isActive

Optional. Defines the resource's status.

Values:

- 1. True
- 0. False

Default: 1

Table and Column: SRM_RESOURCES.Is_Active
Type: Boolean
resourceType
Optional. Defines the resource type.
Values: LABOR, MATERIAL, EQUIPMENT, and EXPENSE
Default: LABOR
Table and Column: SRM_RESOURCES.Resource_type
Type: String

employmentType
Optional. Defines the employment type.
Values: employee and contractor
Default: employee
Table and Column: SRM_RESOURCES.Person_Type
Type: String

hireDate
Optional.
Table and Column: SRM_RESOURCES.Date_of_Hire
Type: Date

terminationDate
Optional.
Table and Column: SRM_RESOURCES.Date_of_Termination
Type: Date

managerUserName
Optional. Identifies the resource's manager.
Table and Column: SRM_RESOURCES.Manager_ID (User Name)
Type: String

isExternal
Optional. Defines the resource's status.
Values:
■ 1. True
■ 0. False
Default: 0
Table and Column: SRM_RESOURCES.Is_External
Type: Boolean
**firstName**
Optional.

*Table and Column:* SRM_RESOURCES.First_Name
*Type:* String

**lastName**
Optional. Validates this field for LABOR resources

*Table and Column:* SRM_RESOURCES.Last_Name
*Type:* String

**middleName**
Optional.

*Table and Column:* SRM_RESOURCES.Middle_Name
*Type:* String

**displayName**
Optional. The name to display on the interface.

*Table and Column:* SRM_RESOURCES.Full_Name
*Type:* String

**emailAddress**
Optional. The email address of the resource.

*Table and Column:* SRM_RESOURCES.Email
*Type:* String

---

**Contact Information Schema Tag (Resources XOG)**

This tag is part schema mapping for the resources XOG object. It has the following attributes:

**companyName**
Optional. The association of a resource to a company.

*Table and Column:* SRM_CONTACTS.Company_ID
*Type:* String

**jobTitle**
Optional.

*Table and Column:* SRM_CONTACTS.Job_Title
*Type:* String
address1
Optional.
**Table and Column:** SRM_CONTACTS.Address1
**Type:** String

address2
Optional.
**Table and Column:** SRM_CONTACTS.Address2
**Type:** String

address3
Optional.
**Table and Column:** SRM_CONTACTS.Address3
**Type:** String

city
Optional.
**Table and Column:** SRM_CONTACTS.City
**Type:** String

state
Optional.
**Table and Column:** SRM_CONTACTS.State_Province
**Type:** String

postalCode
Optional.
**Table and Column:** SRM_CONTACTS.Postal_Code
**Type:** String

countryId
Optional. Lookup values for all countries.
**Table and Column:** SRM_CONTACTS.Country_ID
**Type:** Number

homePhone
Optional. No format enforced.
**Table and Column:** SRM_CONTACTS.Phone_Home
**Type:** String
**workPhone**

Optional. No format enforced.

**Table and Column:** SRM_CONTACTS.Phone_Work

**Type:** String

**mobilePhone**

Optional. No format enforced.

**Table and Column:** SRM_CONTACTS.Phone_Cell

**Type:** String

**fax**

Optional. No format enforced.

**Table and Column:** SRM_CONTACTS.Phone_Fax

**Type:** String

**pager**

Optional. No format enforced.

**Table and Column:** SRM_CONTACTS.Phone_Pager

**Type:** String

**webAddress**

Optional.

**Table and Column:** SRM_CONTACTS.URL

**Type:** String

---

**Management Information Schema Tag**

The Management Information tag is part schema mapping for the resources XOG object. It has the following attributes:

**category**

Optional. Defines the management category for this resource.

**Table and Column:** PRI_RESOURCES.PRCATEGORY

**Type:** String
**defaultAllocationPercentage**
Optional. Defines the default allocation percentage for this resource.

**Default:** 100

**Table and Column:** PRJ_RESOURCES.DEFAULTALLOCATION
**Type:** Number

**trackMode**
Optional. Specifies the method in which this resource tracks time.

**Values:** Clarity, None, and Other.

**Default:** Clarity Time

**Table and Column:** PRJ_RESOURCES.PRTRACKMODE
**Type:** Number

**openForTimeEntry**
Optional. Defines whether this resource is open for time entry.

**Values:**
- 1. True
- 0. False

**Default:** 1

**Table and Column:** PRJ_RESOURCES.PRISOPEN
**Type:** Boolean

**primaryRoleId**
Optional. Indicates the role to which the resource belongs.

**Table and Column:** PRJ_RESOURCES.PRPRIMARYROLEID
**Type:** Number

**inputTypeCode**
Optional. Defines the default input type code for the resource.

**Table and Column:** PRJ_RESOURCES.PRTYPECODEID
**Type:** Number

**userText1**
Optional. A user-defined field.

**Table and Column:** PRJ_RESOURCES.PRUSERTEXT1
**Type:** String
userText3
   Optional. A user-defined field.
   Table and Column: PRJ_RESOURCES.PRUSERTEXT2
type: String

userText3
   Optional. A user-defined field.
   Table and Column: PRJ_RESOURCES.PRUSERTEXT3
type: String

userText4
   Optional. A user-defined field.
   Table and Column: PRJ_RESOURCES.PRUSERTEXT4
type: String

userFlag1
   Optional. A user-defined field.
   Table and Column: PRJ_RESOURCES.PRUSERFLAG1
Type: Boolean

userFlag2
   Optional. A user-defined field.
   Table and Column: PRJ_RESOURCES.PRUSERFLAG2
Type: Boolean

userNumber1
   Optional. A user-defined field.
   Table and Column: PRJ_RESOURCES.PRUSERNUMBER1
Type: Number

userNumber1
   Optional. A user-defined field.
   Table and Column: PRJ_RESOURCES.PRUSERNUMBER1
Type: Number
**Financial Information Schema Tag**

The Financial Information schema tag is part schema mapping for the resource XOG object. It has the following attributes:

**FinancialCode**
- Required. A unique primary key that is equal to the Resource ID. This field is not exposed in the schema.
  - **Table and Column:** PAC_MNT_RESOURCES.RESOURCE_CODE
  - **Type:** String

**location**
- Optional. The location association to the resource.
  - **Table and Column:** PAC_MNT_RESOURCES.LOCATIONID
  - **Type:** String

**department**
- Optional. The department association to the resource.
  - **Table and Column:** PAC_MNT_RESOURCES.DEPARTCODE
  - **Type:** String

**resourceClass**
- Required. The resource class association with the resource.
  - **Table and Column:** PAC_MNT_RESOURCES.RESOURCE_CLASS
  - **Type:** String

**transactionClass**
- Required. The transaction class associated with the resource.
  - **Table and Column:** PAC_MNT_RESOURCES.TRANSCLASS
  - **Type:** String

**vendorCode**
- Optional. The vendor association with the resource.
  - **Table and Column:** PAC_MNT_RESOURCES.VENDOR_CODE
  - **Type:** String
**Active**

Optional. Defines the resource's status.

**Values:**
- 1. True
- 0. False

**Default:** 1

*Table and Column:* PAC_MNT_RESOURCES.ACTIVE

*Type:* Boolean

---

**Expenses Schema Tag**

This tag is part schema mapping for the Resources XOG object. It has the following attributes:

**reimbursementCurrency**

Optional. The lookup value equal to the ISO standard code. This is the currency used to calculate reimbursements made to the resource. This code is not subject to multi-currency rules.

This field is specific to the resource and can vary from the system settings.

*Table and Column:* PAC_MNT_RESOURCES.EX_CURRENCY_CODE

*Type:* String

**employeeCountryCode**

Optional. The reimbursement country code.

*Table and Column:* PAC_MNT_RESOURCES.EX_COUNTRY_CODE

*Type:* String
**Rates and Costs Schema Tag**

The rates and costs schema tag is part schema mapping for the resource XOG object. It has the following attributes:

**targetbillingRate**
- Optional. Defines the target billing rate for this resource.
- **Table and Column:** PAC_MNT_RESOURCES.TARGETBILLRATE
- **Type:** Float

**targetPercentageBillable**
- Optional. Defines the billable percentage for this resource.
- **Table and Column:** PAC_MNT_RESOURCES.TARGETPERCENTBILLABLE
- **Type:** Float

**Custom Information Schema Tag (Resources XOG)**

The custom information tag is part schema mapping for the resource XOG object. It describes custom-defined fields. It has the following attributes:

**Mentor**
- Optional. Identifies the mentor associated with this resource.
- **Table and Column:** XDM_CDF_SRM_RESOURCES.XDM_RESOURCE_MENTOR
- **Type:** String

**WillingTravel**
- Optional. Defines the resource's willingness to travel.
- **Values:**
  - 1. True
  - 0. False
- **Default:** 0
- **Table and Column:** XDM_CDF_SRM_RESOURCES.XDM_WILLING_TRAVEL
- **Type:** Boolean

**ResourceIndustry**
- Optional. Defines the industry related to this resource.
- **Table and Column:** XDM_CDF_SRM_RESOURCES.XDM_RESOURCE_INDUSTRY
- **Type:** String
ResourceGrade
Optional. Defines the grade related to this resource.
Values: Bronze, Silver, Gold, and Platinum.
Table and Column: XDM_CDF_SRM_RESOURCES.XDM_RESOURCE_GRADE
Type: String

OBS Associations Schema Tag (Resources XOG)

The OBS Associations tag is part of the schema mapping for the resources XOG object. It is not mapped to any table or column. This schema tag is a wrapper for the OBSAssoc element.

This schema tag has the following attributes:

completed
Optional. When completed and this value is True, existing OBS associations not listed in the import are deleted.
Default: False
Table and Column: None
Type: String

OBSAssoc Element

This element is associated with the following tables:
- PRJ_OBS_ASSOCIATIONS (OBS Associations)
- PRJ_OBS_TYPES (OBS)
- PRJ_OBS_UNITS (OBS Units)
- PRJ_OBS_UNITS_FLAT (OBS Units Flat Hierarchy)

This element has the following attributes:

id
Required. Defines the OBS association ID.
Table and Column: PRJ_OBS_TYPES.UNIQUE_NAME
Type: String

name
Optional. Defines the OBS association name.
Table and Column: PRJ_OBS_TYPES.PRJ_OBS_TYPES.NAME
Type: String
unitPath

Required. This is a slash-delimited list of unit names leading up to the unit to which the object is associated.

**Table and Column:** PRJ_OBSTYPES.PRJOBS_UNITS.NAME
**Type:** String
**Example:** CAN/BC/VAN

**SkillAssocs Schema Tag (Resources XOG)**

The SkillAssocs tag is part of the schema mapping for the resources XOG object and is a wrapper for the skillAssoc element. It is not mapped to any tables or columns.

This schema tag has the following attributes:

**isComplete**

Optional. If true, this set of skills associations completely replaces any existing skills associated with the resource.

**Default:** False

**Table and Column:** None
**Type:** Boolean

**SkillAssoc Element**

The skills associated with each resource.

**skillAssoc Element**

The skillAssoc element has the following attributes:

**skillCode**

Required. Defines the code for this skill.

**Table and Column:** rsm_skills_associations.skill_id
**Type:** String

**interestLevel**

Optional. Defines the resource's interest level in this skill.

**Table and Column:** CMN_LOOKUPS
**Note:** The lookup code from the cmn_lookups table is based on interest_level_id from the rsm_skills_associations table.
**Type:** String
**proficiencyLevel**

Optional. Defines the resource's proficiency level in this skill.

**Table and Column:** CMN_LOOKUPS

**Note:** The Lookup code from the cmn_lookups table is based on proficiency_level_id from the rsm_skills_associations table.

**Type:** String

**weight**

Optional. Used when searching to produce a weighted average.

**Table and Column:** rsm_skills_associations.weight

**Type:** Number
Resource Class

Use the resource class XOG object to view inbound and outbound resource class instances.

Schema Name

nikuxog_resourceclass.xsd

Read and Write XML Files

The following XML files are included:

- resourceclass_read.xml. Use this file to export resource class instances from CA Clarity.
- resourceclass_write.xml. Use this file to import resource class instances that were previously exported from CA Clarity.

Prerequisites

None.

Read Filters

The following explicit read filters are used:

- resource_class
  Defines the resource class name.
- Resource_type
  Defines the resource class type.
- description
  Defines the description for the resource class.
- active
  Specifies whether the resource class is active.

Error Handling

The following error can be thrown:

- Resource class or description is out of bound.

Schema Mapping

Mappings for the following schema tag name is provided:

- Resource Class
Resource Class (resourceclass) Schema Tag

The resource class tag is part schema mapping for the resource class XOG object. It has the following attributes:

**id**
- Required. Defines the unique resource ID.
- **Table and Column:** PAC_FOS_Resource_Class.ID
- **Type:** String

**resource_class**
- Required. Defines the unique resource class name.
- **Table and Column:** PAC_FOS_Resource_Class.Resource_Class
- **Type:** String

**resource_type**
- Required. Defines the resource type for the resource class.
- **Values:** Q, L, X, and M
- **Table and Column:** PAC_FOS_Resource_Class.RESOURCE_TYPE
- **Type:** String

**description**
- Required. Defines the description of the resource class.
- **Table and Column:** PAC_FOS_Resource_Class.DESCRIPTION
- **Type:** String

**active**
- Required. Defines the status of the resource class.
- **Values:** 0 and 1
- **Table and Column:** PAC_FOS_Resource_Class.ACTIVE
- **Type:** String
Risk

Use the risk XOG object to view inbound and outbound risk attributes associated with projects.

Schema Name

nikuxog_risk.xsd

Read and Write XML Files

The following XML files are included:

- risk_read.xml. Use this file to export risk object instances from CA Clarity.
- risk_write.xml. Use this file to import risk object instances that were previously exported from CA Clarity.

Prerequisites

Before using this XOG, make sure referenced objects, such as projects, users, and categories, exist in CA Clarity.

Read Filters

The following explicit read filters are used:

projectCode
  Defines the code for the associated project.

name
  Defines the name of the change request.

riskCode
  Defines the risk of the change request.

statusCode
  Defines the status of the change request.

priorityCode
  Defines the priority of the change request.

ownerCode
  Defines the name of the owner or assignee of the change request.

Error Handling

The following errors can be thrown:
Risk

Chapter 7: XOG WSDL

- Assessor does not exist in the system.
- Approved By does not exist in the system.
- Project does not exist in the system.
- Category type is not valid.
- Status is not valid.
- Priority is not valid.
- Approach code is not valid.
- Owner does not exist in the system.
- Impact is not valid.
- Probability is not valid.
- Resolved By does not exist in the system.
- Task does not exist for the given project.
- Failed to import risk/issue/change request.

Schema Mapping

Mappings for the following schema tag names are provided:

- risk
Risk Schema Tag

This tag is part of the schema mapping for the risk XOG object, and includes mapping for:

- associatedTasks
- associatedRisks
- associatedIssues
- residualRisks
- responseStrategies

It has the following attributes:

**name**
Required. Defines the risk name.

*Table and Column:* NAME
*Type:* String

**code**
Required. Defines the code for the risk.

*Column and Table:* CODE
*Type:* String

**description**
Defines the description of this risk.

*Table and Column:* DESCRIPTION
*Type:* String

**projectCode**
Defines the code for the associated project

*Table and Column:* INV_INVESTMENTS.CODE
*Type:* String

**categoryTypeCode**
Defines the category of this request.

*Table and Column:* CATEGORY_TYPE_CODE
*Type:* String

**ownerCode**
Defines the name of the owner or assignee of the change request.

*Column and Table:*
Type: String

**statusCode**
Risk status.
**Table and Column:** STATUS_CODE
**Type:** String

**priorityCode**
Defines the priority of the risk (lookup).
**Table and Column:** PRIORITY_CODE
**Type:** String

**assumptions**
Defines the assumptions for this risk.
**Table and Column:** ASSUMPTIONS
**Type:** String

**riskSymptoms**
Risk symptoms.
**Table and Column:** RISK_SYMPTOMS
**Type:** String

**impactDescription**
Description of the impact.
**Table and Column:** IMPACT_DESCRIPTION
**Type:** String

**approachCode**
Response type (lookup)
**Table and Column:** APPROACH_CODE
**Type:** String

**probabilityCode**
Defines the probability code.
**Table and Column:** PROBABILITY_ENUM
**Type:** number

**impactCode**
Impact of this risk (lookup).
**Table and Column:** IMPACT_ENUM
**Type:** String
**Impact Date**

Defines the impact date.

**Table and Column:** IMPACT_DATE  
**Type:** date

**Target Resolution Date**

Targeted date of resolution of this risk.

**Table and Column:** TARGET_RESOLUTION_DATE  
**Type:** Date

**Resolved Date**

Date the risk was resolved.

**Table and Column:** RESOLVED_DATE  
**Type:** Date

**Resolution**

Defines the description or how the risk was resolved.

**Table and Column:** RESOLUTION  
**Type:** String

**Resolved By**

Defines the name of the resource who resolved the risk.

**Table and Column:** RESOLVED_BY  
**Type:** number

---

**Risk Element**

The following attribute is part of the associatedRisks schema tag:

**Code**

Required. Defines the code for the associated risk.

**Table and Column:** CODE  
**Type:** String

---

**Issue Element**

The following attribute is part of the associatedIssues schema tag:

**Code**

Required. Defines the code for the associated issue.

**Table and Column:** CODE
**responseStrategy Element**

The following attributes are part of the responseStrategies schema tag:

- **description**
  
  Description of this risk response strategy.
  
  **Column and Table:** DESCRIPTION
  
  **Type:** String

- **assignedTo**

  Defines the name of the resource assigned to this risk.
  
  **Column and Table:** ASSIGNED_TO
  
  **Type:** String

- **resolveBy**

  Targeted date of resolution of this risk.
  
  **Column and Table:** TARGET_RESOLUTION_DATE
  
  **Type:** Date

**Role**

Use the role XOG object to view inbound and outbound role attributes.

**Schema Name**

nikuxog_role.xsd

**Read and Write XML Files**

The following XML files are included:

- **roles_read.xml.** Use this file to export role attributes from CA Clarity.
- **roles_write.xml.** Use this file to import role attributes that were previously exported from CA Clarity.
**Business Rules and Processing**

The roles schema is defined for both inbound (write) and outbound (read) processing.

**Important!** Set up the following items correctly before importing resources:

**Financial Properties**

The financial properties that are associated with the role.

Before you can import roles into the product, set up the following financial properties in the Financial Administration module. The *required* property values must be present; otherwise, the financial properties are not imported and an error is written to the Success and Error File.

**Transaction Class**

Required if the role is financially active. The transaction class that is associated with the role. The value in the XOG can be found in the TRANSCLASS table.

**Resource Class**

Required if the role is financially active. The resource class that is associated with the role. The value in the XOG can be found in the PAC_FOS_RESOURCE_CLASS table.

**Vendor**

Optional: The vendor that is associated with the role. This property is not required for the financial properties. If the vendor property value is missing, the role is added without a vendor property and a warning is posted to the Success and Error file.

**Read Filters**

The XOG allows or outbound processing of roles that are based on the following fields:

- active. The possible values for this field are: Active and Inactive.
- resourceId. The resourceId corresponds to unique_name in SRM_RESOURCES.

**Error Handling**

If the parent role or the standard calendar does not exist, XOG displays an error message and does not import or update the record.
**Schema Mappings**

The following tables and schema tags are mapped to roles:

- PRJ_RESOURCES table
- SRM_RESOURCES table
- PRJ_ROLES_FLAT table
- PAC_FOSRESOURCE_CLASS table
- SkillsAssocs schema tag

**PRJ_RESOURCES Schema Tag**

This tag is part schema mapping for the role XOG object. It has the following attributes:

**category**

Optional.

**Table and Column:** PRJ_RESOURCES.PRCategory

**Type:** String

**availability**

Required. This attribute is stored as a blob internally. It uses standard calendar for conversion.

**Table and Column:** PRJ_RESOURCES.PRAvailCurve

**Type:** Double
SRM.getResources Schema Tag

This tag is part schema mapping for the role XOG object. It has the following attributes:

name
   Required. Must be unique.
   Table and Column: SRM_RESOURCES.Last_Name,Full_Name
   Type: String

resourceId
   Required.
   Table and Column: SRM_RESOURCES.Unique_Name
   Type: String

active
   Optional.
   Table and Column: SRM_RESOURCES.Is_Active
   Type: Boolean

PRJ.ROLES_FLAT Schema Tag

This tag is part schema mapping for the role XOG object. It has the following attribute:

parentRole
   Optional.
   Table and Column: PRJ/Resources.Branch_Role_Id
   The ID of the parent role is stored as the branch_role_id.
   Type: String
Financial Information Schema Tag

The Financial Information schema tag is part schema mapping for the role XOG object. It has the following attributes:

**Active**

Required. Defines the status of the role.

**Values:**
- 1. True
- 0. False

**Table and Column:** PAC_MNT_RESOURCES.ACTIVE

**Type:** Boolean

**resourceClass**

Required. The resource class association with the role.

**Table and Column:** PAC_MNT_RESOURCES.RESOURCE_CLASS

**Type:** String

**transactionClass**

Required. The transaction class associated with the role.

**Table and Column:** PAC_MNT_RESOURCES.TRANSCLASS

**Type:** String

**vendorCode**

Optional. The vendor association with the role.

**Table and Column:** PAC_MNT_RESOURCES.VENDOR_CODE

**Type:** String
**SkillAssocs Schema Tag**

The SkillAssocs tag is part schema mapping for the role XOG object. This is a wrapper for the individual Skills Associations for a role.

The SkillAssocs schema tag has the following attribute:

**isComplete**

Optional. If true, this set of skills associations completely replaces any existing skills associated with the role.

**Default:** False

**Table and Column:** None

**Type:** Boolean

**SkillAssoc (Skills Association) Element**

This represents the skills associated with each role.

**skillCode**

Required. The code for the skill.

**Table and Column:** rsm_skills_associations.skill_id

**Type:** String

**interestLevel**

Optional. The role's interest level in this skill.

**Table:** CMN_LOOKUPS

The Lookup code from the cmn_lookups table is based on the interest_level_id from the rsm_skills_associations table.

**Type:** String

**proficiencyLevel**

Optional. The role's proficiency level in this skill.

**Table:** CMN_LOOKUPS

The Lookup code from the cmn_lookups table is based on the proficiency_level_id from the rsm_skills_associations table.

**Type:** String

**weight**

Optional. Used when searching to produce a weighted average.

**Table and Column:** rsm_skills_associations.weight

**Type:** Number
Skill

Use the skill XOG object to view inbound and outbound skill object instances.

Schema Name

nikuxog_skill.xsd

Read and Write XML Files

The following XML files are included:

- rsm_skills_read.xml. Use this file to export skill object instances from CA Clarity.
- rsm_skills_write.xml. Use this file to import skill object instances that were previously exported from CA Clarity.

Prerequisites

None

Read Filters

The following explicit read filters are used:

- skillName
- isActive

Error Handling

A basic import failure error can be thrown.

Schema Mapping

Mappings for the following schema tag names are provided:

- Skill
Skill Schema Tag

The skill tag is part of the schema mapping for the skill XOG object. It has the following attributes:

**isActive**
- Defines the status of the skill.
- **Table and Column:** IS_ACTIVE
- **Type:** Boolean

**description**
- Defines the description of skill.
- **Table and Column:** DESCRIPTION
- **Type:** String

**skillCode**
- Required. Defines the unique identifier for the skill.
- **Table and Column:** SKILL_CODE
- **Type:** String

**name**
- Defines the skill name.
- **Table and Column:** SKILL_NAME
- **Type:** String
Subproject (Program)

Use the subproject (program) XOG object for inbound (write) and outbound (read) processing of subproject and program data. Subprojects are the links between master projects and the projects they contain.

Schema Name

nikuxog_project.xsd

Read and Write XML Files

The following XML files are included:

- prj_programs_read.xml. Use this file to export programs from CA Clarity.
- prj_programs_write.xml. Use this file to import programs that were previously exported from CA Clarity.

Business Rules and Processing

The following business rules and processing apply to this XOG:

- When you import a subproject, data for the subproject is:
  - Updated (if it has already been imported).
  - Created (if it has not already been imported).
- The only field supported during an update is read-only. To change the content of other fields during an update, delete the existing sub-project and create a new one (all fields are refreshed, not just the read-only field).
- When you create a subproject, a proxy task is also created in the WBS of the master project that serves as a place holder for the sub-project. When you remove the subproject, the proxy is also removed.
- When you import a proxy task, the WBSSequence for all subsequent tasks is incremented by 1.

Read Filters

To allow for outbound processing of programs, the XOG adds the isProgram filter to projects. This filter performs the following actions. When:

- isProgram is commented out, all appropriate programs and projects are exported.
- isProgram=1, programs only are exported.
- isProgram=0, projects only are exported.

Schema Mappings
The prSubProject table contains information about subprojects. The data it contains is primarily linking information that specifies if the subproject is partial or complete. The link also specifies if the subproject is read-only. By definition, partial subprojects are read-only.

The following schemas are described:
- PRSubproject Schema Tag (Inbound and Outbound) (see page 400)
- PRSubproject Schema Tag (Inbound only) (see page 401)

**PRSubproject Schema Tag (Inbound and Outbound)**

This tag is part schema mapping for the PRSub XOG object. It has the following attributes:

- **projectID**
  - Required. The UNIQUE_NAME of the project that becomes the sub-project of the project when imported.
  - **Table and Column:** PRSubproject.prRefProjectID
  - **Type:** String

- **TaskID**
  - Optional. The prExternalID of the task that represents the portion of the Work breakdown structure (WBS) that becomes the sub-project of the project. This can be defined at any level of the WBS.
  - **Table and Column:** PRSubproject.prRefTaskID
  - **Type:** String

- **succeedingTaskID**
  - Optional. The prExternalID of the task that follows the sub-project in the WBS. If the sub-project is (or should be) the last item in the WBS, the value is not present.
  - **Table and Column:** None
  - **Type:** String

- **ReadOnly**
  - Optional. This specifies if changes made to the master project are to be persisted when the master project is saved from Open Workbench or Microsoft Project. For partial sub-projects, this value is always False.
  - **Table and Column:** PRSubproject.prReadOnly
  - **Type:** Boolean
PRSubproject Schema Tag (Inbound only)

This tag is part schema mapping for the PRSubp XOG object. This tag is associated with the PRSubproject table. It has the following attributes:

**delete**

Optional. When this attribute is present on an inbound transaction, the sub-project link is deleted with the associated proxy.

**Table and Column:** None

**Type:** Boolean
Subscription

Use the subscription XOG object to view inbound and outbound department subscription attributes.

Schema Name

nikuxog_subscription.xsd

Read and Write XML Files

The following XML files are included:

- subscription_read.xml. Use this file to export department subscription attributes from CA Clarity.

- subscription_write.xml. Use this file to import department subscription attributes that were previously exported from CA Clarity entity, department and service must exist in CA Clarity.

Read Filters

The following explicit read filters are used:

departmentId

The code of the department for which the subscriptions should be read out.

Error Handling

The errors are thrown based on the following checks:

- Required fields. Ensures all required fields have values.
- Entity. Checks if the entity is valid and exists.
- Department. Checks if the department is valid and exists.
- Service. Checks if the service is valid and exists.

Schema Mapping

Mappings for the following schema tag name is provided:

- Subscription (see page 403)
Subscription Schema Tag

The subscription tag is part of the schema mapping for the subscription XOG object. This is a placeholder tag for multiple subscriptions.

Subscription element

There can be zero or more subscription elements each having an optional keymetrics element, Following are the attributes of a subscription element:

- **sla_violations**
  - Defines the number of SLA violations.
  - **Table and Column**: DPT_SUBSCRIPTIONS.sla_violations
  - **Type**: Integer

- **sla_violations_th**
  - Defines the threshold for SLA violations.
  - **Table and Column**: DPT_SUBSCRIPTIONS.sla_violations_threshold
  - **Type**: Integer

- **incidents**
  - Defines the number of incidents.
  - **Table and Column**: DPT_SUBSCRIPTIONS.incidents
  - **Type**: Integer

- **incidents_threshold**
  - Defines the threshold for incidents.
  - **Table and Column**: DPT_SUBSCRIPTIONS.incidents_threshold
  - **Type**: Integer

- **change_orders**
  - Defines the number of change orders.
  - **Table and Column**: DPT_SUBSCRIPTIONS.change_orders
  - **Type**: Integer

- **charges**
  - Defines the total charges (from chargebacks) against the investment (service) for this subscription.
  - **Table and Column**: DPT_SUBSCRIPTIONS.charges
Subscription

Type: Integer
cust_satisfaction
Defines the customer satisfaction rating for this subscription.
Table and Column: DPT_SUBSCRIPTIONS.customer_satisfaction

Type: Integer
total_users
Defines the total number of users utilizing this subscription.
Table and Column: DPT_SUBSCRIPTIONS.total_users

Type: Integer
active_users
Defines the number of active users utilizing this subscription.
Table and Column: DPT_SUBSCRIPTIONS.active_users

Type: Integer
page_hits
Defines the page hits as captured for this subscription if applicable.
Table and Column: DPT_SUBSCRIPTIONS.page_hits

entityId
Required. Defines the entity to which the service belongs.
Table and Column: This is a derived attribute.

Type: String
departmentId
Required. Identifies the subscribing department.
Table and Column: DPT_SUBSCRIPTIONS.department_id

Type: String
serviceId
Required. Defines the identifier that makes it unique in combination with the table_name column.
Table and Column: pk_id
keymetrics Tag

The keymetrics element consists of zero or more keymetric elements. This element contains an optional targetCurve and an actualCurve and may have zero or more CustomInformation elements. It has the following attributes:

metrics_code
  Required. Code of the metric.

Table and Column: DPT_KEYMETRICS.METRIC_CODE
  Type: String

metrics_code
  Optional.

Table and Column: DPT_KEYMETRICS.NAME
  Type: String

targetCurve and actualCurve Schema Tag

The curve elements contain segment objects which specify target metrics and actual metrics over a period of time.
**Time Period**

Use the time period (timesheet) XOG object to view inbound and outbound timesheet attributes. Timesheet information includes time periods and resources assigned to tasks. You can export the data to serve the purposes of system integration.

For example, you might do this when you need the information to drive internal or external billing systems. You do not need to export all timesheets in a system nor do you need to export timesheets individually. Filtering options are therefore provided by the XOG to streamline the export requests.

**Schema Name**

nikuxog_timeperiod.xsd

**Read and Write XML Files**

The following XML files are included:

- olt_timeperiods_read.xml. Use this file to export time period instances from CA Clarity.

**Read Filters**

The XOG processes outbound capacity planning schemas based on the following fields:

- **isPublic**
  - 1 or 0

- **ownerID**
  - Valid user (CMN_SEC_USERS.ID)

The XOG uses no implicit filters for timesheets (timesheets for non-labor resources are excluded). The explicit filters used are:

- Start Date
- resourceID
- postedInTimePeriodStart

**Schema Mappings**

Mappings for the following schema tags are provided:

- TimePeriod
- PRTimesheet
- SRM_RESOURCES
- PRTimePeriod
<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRTimeEntry</td>
<td></td>
</tr>
<tr>
<td>PRChargeCode</td>
<td></td>
</tr>
<tr>
<td>PRTTypeCode</td>
<td></td>
</tr>
<tr>
<td>PRJ_Projects</td>
<td></td>
</tr>
<tr>
<td>PRTask</td>
<td></td>
</tr>
<tr>
<td>PRAssignments</td>
<td></td>
</tr>
<tr>
<td>NoteData</td>
<td></td>
</tr>
</tbody>
</table>

**TimePeriod Schema Tag**

This tag is part schema mapping for the time period XOG object. It has the following attributes:

- **start**
  - Required.
  - **Table and Column:** PRTimePeriod.prStart
  - **Type:** DateTime

- **finish**
  - Required.
  - **Table and Column:** PRTimePeriod.prFinish
  - **Type:** DateTime

- **openForTimeEntry**
  - Optional.
  - **Table and Column:** PRTimePeriod.prIsOpen
  - **Type:** Boolean

- **postedTime**
  - Optional.
  - **Table and Column:** PRTimePeriod.prPostedTime
  - **Type:** DateTime
**PRTimesheet Schema Tag**

This tag is part schema mapping for the time period XOG object. It has the following attributes:

**ID**
- Required. This is an internally-generated unique identifier.
- **Table and Column:** PRTimeSheet.prID
- **Type:** Integer

**status**
- Optional.
- **Values:** Unsubmitted, Submitted, Rejected, Approved, and Posted
- **Table and Column:** PRTimeSheet.prStatus
- **Type:** Integer

**submittedBy**
- Optional.
- **Table and Column:** PRTimeSheet.prSubmittedBy
- **Type:** String

**approvedBy**
- Optional.
- **Table and Column:** PRTimeSheet.prApprovedBy
- **Type:** String

**adjustedTimeSheetID**
- Optional. This refers to the ID attribute of the timesheet that this timesheet is adjusted by, if any.
- **Table and Column:** PRTimeSheet.prAdjustedId
- **Type:** Integer
**SRM/Resources Schema Tag**

The SRM/Resources tag is part schema mapping for the time period XOG object. It has the following attributes:

This tag uses the SRM/Resources table where SRM/Resources.ID = PRTimesheet.prResourceID.

**resourceID**

Required.

*Table and Column:* SRM/Resources.UNIQUE_NAME

*Type:* String

**PRTimePeriod Schema Tag**

This tag is part schema mapping for the time period XOG object. This tag uses the PRTimePeriod table where PRTimePeriod.ID = PRTimesheet.prPostedPeriodID. It has the following attributes:

**postedInTimePeriodStart**

Optional. The time period in which the timesheet was posted. This may differ from the time period in which this attribute is found if the timesheet was approved late, as with an adjustment.

*Table and Column:* PRTimePeriod.prStart

*Type:* dateTime

**PRTimeEntry Schema Tag**

The PRTimeEntry tag is part schema mapping for the time period XOG object. This tag uses the PRTimeEntry table, 0 to many where PRTimeEntry.prTimeSheetID = PRTimesheet.prID. It has the following attributes:

**totalActuals**

Optional. The total actuals in hours.

*Table and Column:* PRTimeEntry.prActSum

*Type:* float
**taskID**

Optional. For existing assignments: you can use projectID and taskID; or assignmentID.

For new assignments: you must use projectID and internalTaskID.

**Table and Column:** prTask.prExternalID

**Type:** string

**projectID**

Optional. For existing assignments: you can use projectID and taskID; or assignmentID.

For new assignments: you must use projectID and internalTaskID.

**Table and Column:** srm_projects.unique_name

**Type:** string

**internalTaskID**

Optional. For existing assignments: you can use projectID and taskID; or assignmentID.

For new assignments: you must use projectID and internalTaskID.

**Table and Column:** srm_projects.prTask.prID

**Type:** Integer

**assignmentID**

Optional. You can achieve a timesheet entry without creating an assignment first, but you must specify the task by its internal numeric ID number, such as internalTaskID = "5000876". If an assignment exists, you can use taskID = "abc".

**Table and Column:** prAssignment.prID

**Type:** Integer

**actualDate**

Required. Zero to many records for each day during this time period for which this time entry has actuals.

**Table and Column:** None

**Type:** Date

**amount**

Optional. The data is entered as hours. This is the amount for the date only.

**Table and Column:** prActCurve

**Type:** Float
PRTTypeCode Schema Tag

The PRTTypeCode tag is part schema mapping for the time period XOG object. This tag uses the PRTTypeCode table where PRTTimeEntry.prTypeCodeID = PRTTypeCode.prID. It has the following attributes:

**typeCodeID**
- Optional.
- **Table and Column:** PRTTypeCode.prExternalID
- **Type:** String

**typeCodeName**
- Optional.
- **Table and Column:** PRTTypeCode.prName
- **Type:** String

PRChargeCode Schema Tag

This tag is part schema mapping for the time period XOG object. It has the following attributes:

This tag uses the PRChargeCode table where PRTTimeEntry.prChargeCodeID = PRChargeCode.prID.

**ChargeCodeID**
- Optional.
- **Table and Column:** PRChargeCode.prExternalID
- **Type:** String

**ChargeCodeName**
- Optional.
- **Table and Column:** PRChargeCode.prName
- **Type:** String
**PRJ_Projects Schema Tag**

This tag is part schema mapping for the time period XOG object.

The following is true regarding the attributes that follow this statement:
From PRJ_PROJECTS, where PRTimeEntry.prAssignmentID = PRAssignment.prID and PRAssignment.prTaskID = PRTask.prID and PRTask.prProjectID = PRJ_PROJECTS.prID.

When prAssignmentID is not set (as for indirect time entries), these fields are not included.

**projectID**

Optional.

*Table and Column:* PRJ_PROJECTS.UNIQUE_NAME

*Type:* String

**projectName**

Optional.

*Table and Column:* PRJ_PROJECTS.NAME

*Type:* String

**PRTask Schema Tag**

This tag is part schema mapping for the time period XOG object.

The following is true regarding the attributes that follow this statement:
- This tag uses the PRTask table where PRTimeEntry.prAssignmentID = PRAssignment.prID and PRAssignment.prTaskID = PRTask.prID).
- When prAssignmentID is not set (as for indirect time entries), these fields are not included.

This tag includes the following attributes:

**taskID**

Optional.

*Table and Column:* PRTask.prExternalID

*Type:* String

**taskName**

Optional.

*Table and Column:* PRTask.prName

*Type:* String
PRAssignments Schema Tag

The PRAssignments tag is part schema mapping for the time period XOG object.

The following is true regarding the attributes that follow this statement:
- This tag uses the PRAssignments table where PRTimeEntry.prAssignmentID = PRAssignment.prID.
- When prAssignmentID is not set (as for indirect time entries), these fields are not included.

This tag includes the following attributes:

- **assignmentStart**
  - Optional.
  - **Table and Column:** PRAssignments.prStart
  - **Type:** dateTime

- **assignmentFinish**
  - Optional.
  - **Table and Column:** PRAssignments.prFinish
  - **Type:** dateTime

- **assignmentPendingEstimates**
  - Optional. The value is in hours.
  - **Table and Column:** PRAssignments.prPendEstSum
  - **Type:** Float

- **assignmentEstimate**
  - Optional. The value is in hours.
  - **Table and Column:** PRAssignments.prEstSum
  - **Type:** Float

- **assignmentEstimateForTimePeriod**
  - Optional. The value is in hours, where the total within this time period only
  - **Table and Column:** PRAssignments.prEstCurve
  - **Type:** Float
NoteData Schema Tag

The NoteData tag is part schema mapping for the time period XOG object. This tag uses the PRNote, which is 0 to many, and where PRNote.prRecordID = PRTimesheet.prID and PRNote.prTableName = 'PRTimesheet'. These values are contained in the TimeSheet Notes tag.

category

Optional. Defines the category for the timesheet note.

**Table and Column:** NoteData.prCategory
**Type:** String

noteText

Optional. Defines the text for the timesheet note.

**Table and Column:** NoteData.prValue
**Type:** String

createdBy

Optional. Defines the name of the resource who created the timesheet note.

**Table and Column:** NoteData.prCreatedBy
**Type:** String

createdTime

Optional. Defines the date the timesheet note was created.

**Table and Column:** NoteData.prCreatedTime
**Type:** dateTime
Transaction Class

Use the transaction class XOG object to view inbound and outbound transaction class instances.

Schema Name

nikuxog_transactionclass.xsd

Read and Write XML Files

The following XML files are included:
- transactionclass_read.xml. Use this file to export transaction class instances from CA Clarity.
- transactionclass_write.xml. Use this file to import transaction class instances that were previously exported from CA Clarity.

Prerequisites

None.

Read Filters

The following explicit read filters are used:
- transclass
  Defines the transaction class name.
- transtype
  Defines the transaction class type.
- description
  Defines the transaction class descriptions.
- shortdesc
  Defines the summary of the transaction class description.

Error Handling

The following errors can be thrown:
- Description or short description is out of bound.

Schema Mapping

Mappings for the following schema tag name is provided:
- Transaction Class (see page 416)
Transaction Class (transactionclass) Schema Tag

The transaction class tag is part schema mapping for the transaction class XOG object. It has the following attributes:

**transactionclass**
- Required. Defines the unique transaction class name.

  **Table and Column:** TRANSACTIONCLASS
  **Type:** String

**transactiontype**
- Required. Defines the transaction type for the transaction class.

  **Values:** Q, L, X, and M

  **Table and Column:** RESOURCE_TYPE
  **Type:** String

**description**
- Required. Defines the description of the transaction class.

  **Table and Column:** DESCRIPTION
  **Type:** String

**shortdesc**
- Required. Defines the summary of the transaction class description.

  **Table and Column:** SHORTDESC
  **Type:** String
Type Code

Use the type code XOG object to view inbound and outbound type code object instances.

Schema Name

nikuxog_typecode.xsd

Read and Write XML Files

The following XML files are included:
- prj_typecodes_read.xml. Use this file to export type codes from CA Clarity.
- prj_typecodes_write.xml. Use this file to import type codes that were previously exported from CA Clarity.

Prerequisites

None

Read Filters

The following explicit read filter is used:
- open

Error Handling

A basic import failure error can be thrown.

Schema Mapping

Mappings for the following schema tag name is provided:
- Type Code

Type Code Schema Tag

The Type Code XOG is composed of the TypeCode element, which has the following attributes:

Id

Defines the unique identifier for the type code.

Table and Column: prID

Type: Integer
**Type Code**

- **typeCodeID**
  - Required. Defines the external unique identifier for the type code.
  - **Table and Column:** prExternalID
  - **Type:** String

- **name**
  - Required. Defines the name of the type code.
  - **Table and Column:** prName
  - **Type:** String

- **openForTimeEntry**
  - Defines whether the type code is open for timesheet use.
  - **Table and Column:** prIsOpen
  - **Type:** BOOLEAN

- **isChargeable**
  - Defines whether the type code is chargeable in financial systems.
  - **Table and Column:** IS_CHARGEABLE
  - **Type:** BOOLEAN
UI Themes

Use the UI Themes XOG object for outbound (read) and inbound (write) processing.

Schema Name

nikuxog_uitheme.xsd

Read and Write XML Files

The following files are included.
- cmn_ui_themes_read.xml. Use this file to export UI themes.
- cmn_ui_themes_write.xml. Use this file to import UI themes that were previously exported.

Prerequisites

None

Business Rules and Processing

The last UI theme that is imported with an attribute value of default=true is the default UI theme for the system.

Read Filters

The following explicit read filter is used:

uiThemeID

Defines the unique UI theme ID that must be read out.

Schema Mappings

The following schema mappings are described:
- UI Theme (see page 420)
- NLS (see page 420)
- CSS (see page 421)
UI Theme Schema Tag

The UITheme schema tag is part of schema mapping for the UI Theme XOG object. The schema tag has the following attributes:

**id**
Required. Defines the unique ID of the UI theme.

*Table and Column:* CMN_UI_THEMES.CODE
*Type:* String

**active**
Required. Defines whether the UI theme is active.

*Table and Column:* CMN_UI_THEMES.IS_ACTIVE
*Type:* String
*Default:* True

**default**
Optional. Defines whether the UI theme is the default UI theme.

*Table and Column:* CMN_UI_THEMES.IS_DEFAULT
*Type:* String
*Default:* False

NLS Schema Tag

The UITheme tag is part of schema mapping for the UI Theme XOG object. It has the following attributes:

**Name**
Required. Defines the name of the UI theme.

*Table and Column:* CMN_CAPTIONS_NLS.NAME
*Type:* NlsType

**Description**
Required. Provides a description of the UI theme.

*Table and Column:* CMN_CAPTIONS_NLS.DESCRIPTION
*Type:* NlsType
CSS Schema Tag

This required schema tag defines the CSS code of the UI theme. The content of this element should always be enclosed in a CDATA section so that CSS code is not part of the markup of this schema.
User

Use the user XOG object to view inbound and outbound user object instance attributes.

**Schema Name**

nikuxog_user.xsd

**Read and Write XML Files**

The following XML files are included:

- cmn_users_read.xml. Use this file to export users from CA Clarity PPM.
- cmn_users_write.xml. Use this file to import users that were previously exported from CA Clarity PPM.

**Business Rules and Processing**

Users are defined for both inbound (write) and outbound (read) processing. Password and Password_Confirm, used to validate the user, are not exposed but are populated with default values (2000). When a user first logs in, they are prompted to reset this default password.

**Resource**

A labor resource is automatically created for every user imported through XOG.

**Company ID**

A browse field used to associate a user to a company is run against SRM_COMPANIES. If the company does not exist, the user is posted without a company and a warning message is posted to the Success and Error file. If the company_id exists, the field is populated with that value.

**Lookup values**

The schema requires lookup codes that are validated against CMN_LOOKUPS.

**User type**

If not provided, this is defaulted to internal. There is no admin type.

**OBS association**

With the new OBS Security, a Security OBS is required and any OBS can be associated with a user. To accommodate this, there is an OBS Associations portlet. The OBS association fields can be used for import and export.

**Read Filters**

This XOG allows for outbound (read) processing of users based on the following two fields: User Status and User Type. And and Or processing is supported between these two fields and for processing within Type. The following combinations are supported:
User Status = x
where x = Active, Inactive, or LOCK

User Type = x
where x = Internal, External

User Status = x AND User Type = y
where x = Active, Inactive, or LOCK
where y = one of many User Types

**Error Handling**

If an error occurs for a user transaction, the following information is written to the Success and Error file:

- externalId
- ExternalSource

**Schema Mappings**

The following schema mappings are described:

- Personal Information (CMN_SENC_USERS)
- OBS Associations (OBSAssocs)
- Group Assignments
- Global Access Right Assignments (GlobalRights)
- Instance Access Right Assignments (InstanceRights)
- Instance OBS Access Right Assignments (InstanceOBSRights)
- Instance Object (InstanceObject)
- Language Support (nls)

**Personal Information (CMN_SENC_USERS) Schema Tag**

This tag is part schema mapping for the user XOG object. It has the following attributes:

- **firstName**
  - Required.
  - **Table and Column:** CMN_SEC_USERS.FirstName
  - **Type:** String
**lastName**

Required.

*Table and Column:* CMN_SEC_USERS.Last Name  
*Type:* String

**userName**

Required. A unique primary key.

*Table and Column:* CMN_SEC_USERS.User_Name  
*Type:* String

**userType**

Optional.

*Values:* Internal and External  
*Default:* Internal

*Table and Column:* CMN_SEC_USERS.User_Type_Id  
*Type:* String

**userStatus**

Required.

*Values:* Active, Inactive, and LOCK  
*Default:* Active

*Table and Column:* CMN_SEC_USERS.User_Status_ID  
*Type:* String

**emailAddress**

Required.

*Table and Column:* CMN_SEC_USERS.Email Address  
*Type:* String

**userLocale**

Optional. The Java Locale format, for example, en_US.

*Table and Column:* CMN_SEC_USERS.locale  
*Type:* String

**userTimezone**

Optional. The Java TimeZone format, for example, Europe/London, PST.

*Table and Column:* CMN_SEC_USERS.timezone  
*Type:* String
**userLanguage**
Required. Defines the language displayed when the user first logs in.

*Values*: English, German, Spanish, and French

*Default*: English

*Table and Column*: CMN_SEC_USERS.Language

*Type*: String

**resource**
Optional. A browse field with a one-to-one relationship between users and resources.

*Table and Column*: CMN_SEC_USERS.Resource

*Type*: String

**companyIId**
Optional. This is a browse field. The company association with the user.

*Table and Column*: CMN_SEC_USERS.Company_ID

*Type*: String

**externalSource**
Required by the schema. It is a lookup value that is the originating system ID (for example, Oracle).

*Table and Column*: CMN_SEC_USERS.External_Source_ID

*Type*: String (in schema) and Number (in the application)

**externalId**
Required by the XML schema. The originating unique identifier.

*Table and Column*: CMN_SEC_USERS.External_ID

*Type*: String
OBS Associations (OBSAssocs) Schema Tag

This tag is part schema mapping for the user XOG object. This is a wrapper for the OBSAssoc element. It is not mapped to any table.

The OBS Associations schema tag has the following attribute:

completed

Optional. When completed and this value is True, the existing OBS associations not listed in the import are deleted.

Default: False

Table and Column: None

Type: String

OBS Association (OBSAssoc) Element

This element is mapped to the following tables:

- PRJ_OBS_ASSOCIATIONS (OBS Associations)
- PRJ_OBS_TYPES (OBS)
- PRJ_OBS_UNITS (OBS Units)
- PRJ_OBS_UNITS_FLAT (OBS Units Flat Hierarchy)

id

Required.

Table and Column: PRJ_OBS_TYPES.UNIQUE_NAME

Type: String

name

Optional.

Table and Column: PRJ_OBS_TYPES.PRJ_OBS_TYPES.NAME

Type: String

unitPath

Required. This is a slash-delimited list of unit names leading up to the unit to which the object is associated.

Table and Column: PRJ_OBS_TYPES.PRJ_OBS_UNITS.NAME

Type: String

Example: CAN/BC/VAN
**Group Assignments Schema Tag**

This tag is part schema mapping for the user XOG object. It is a wrapper for the Group elements.

The Group Assignments schema tag is not mapped to any table. It has the following attribute:

- **completed**
  - Optional. If completed and set to True, any existing Group assignments that are not listed in the import are deleted.
  - **Default**: False
  - **Table and Column**: none
  - **Type**: String

**Group Assignment (Group Assignments) Element**

This is a wrapper element for the Group elements. There can be many Group elements. It has the following attribute:

- **id**
  - Required.
  - **Table and Column**: CMN_SEC_GROUPS.GROUP_CODE
  - **Type**: String
Global Access Right Assignments (GlobalRights) Schema Tag

This tag is part schema mapping for the user XOG object. It is a wrapper element for the Right elements. There can be many Right elements.

The Global Access Right Assignments schema tag is not mapped to any table. It has the following attribute:

completed
Optional. If completed and set to True, then existing Right assignments not listed in the import are deleted.

Default: False
Table and Column: None
Type: String

Right Assignment (Right) Element

id
Required.

Table and Column: CMN_SEC_GROUPS.GROUP_CODE
Type: String
Instance Access Right Assignments (InstanceRights) Schema Tag

This tag is part schema mapping for the user XOG object. It is a wrapper element for the Right elements. There can be many Right elements.

The Instance Access Right Assignments schema tag is not mapped to any table. It has the following attribute:

completed
  Optional. If completed and set to True, any existing Right assignments not listed in the import are deleted.
  
  **Default:** False
  
  **Table and Column:** None
  
  **Type:** String

Right Assignment (Right) Element

id
  Required.
  
  **Table and Column:** CMN_SEC_GROUPS.GROUP_CODE
  
  **Type:** String
Instance OBS Access Right Assignments (InstanceOBSRights) Schema Tag

This tag is part schema mapping for the user XOG object. It is a wrapper element for the Right elements. There can be many Right elements.

The Instance OBS Access Right Assignments schema tag is not mapped to any table. It has the following attribute:

completed
  Optional. If completed and set to True, any existing Right assignments not listed in the import are deleted.
  Default: False
  Table and Column: None
  Type: String

Right Assignment (Right) Element

id
  Required.
  Table and Column: CMN_SEC_GROUPS.GROUP_CODE
  Type: String
**Instance Object (InstanceObject) Schema Tag**

This tag is part schema mapping for the user XOG object. It has the following attributes that are mapped to any of the following tables, unless otherwise noted:

- SRM_RESOURCES
- SRM_PROJECTS
- BPM_DEF_PROCESSES
- CMN_PAGES
- CMN_PORTLETS
- CMN_SCH_JOB_DEFINITIONS
- INV.APPLICATION
- INV_ASSET
- INV_OTHER
- INV_PRODUCT
- SCENARIO

**id**

Required. The unique code from one of the listed tables.

**Type:** String

**name**

Optional. The name from one of the listed tables.

**Type:** String

**type**

Required. The key to determine which table is mapped. This is not mapped to any table.

**Type:** String

---

**Language Support (nls) Schema Tag**

This tag is part schema mapping for the user XOG object. It has the following attributes:

**name**

Optional.

**Table and Column:** CMN_CAPTIONS_NLS.NAME

**Type:** String
description
Optional.
Table and Column: CMN_CAPTIONS_NLS.DESCRIPTION
Type: String

languageCode
Optional.
Table and Column: CMN_CAPTIONS_NLS.LANGUAGE_CODE
Type: String
Vendor

Use the Vendor XOG object to view inbound and outbound vendor instances.

Schema Name

nikuxog_vendor.xsd

Read and Write XML Files

The following XML files are included:
- vendor_read.xml. Use this file to export Vendor instances from CA Clarity.
- vendor_write.xml. Use this file to import Vendor instances that were previously exported from CA Clarity.

Prerequisites

None

Read Filters

The following explicit read filters are used:
- code
  Defines the unique vendor code.
- name
  Defines the name of the vendor.
- status
  Defines the status of the vendor.

Error Handling

The following errors can be thrown:
- Could not xog-in item because size of some attributes [description] is not within valid range.

Schema Mapping

Mappings for the following schema tag name is provided:
- Vendor (see page 434)
Vendor Schema Tag

The Vendor tag is part schema mapping for the Vendor XOG object. It has the following attributes:

**code**
- Required. Defines the unique vendor code.
  - *Table and Column:* VENDOR_CODE
  - *Type:* String

**name**
- Required. Defines the name of the vendor.
  - *Table and Column:* ADDRESS_NAME
  - *Type:* String

**affiliation**
- Optional.
  - *Table and Column:* AFFILIATED_VEND_CODE
  - *Type:* String

**status**
- Required. Defines the status of the vendor.
  - *Valid Values:* Active, Inactive, and No New Business
  - *Table and Column:* STATUS_TYPE
  - *Type:* String

**address1**
- Optional.
  - *Table and Column:* ADDR1
  - *Type:* String

**address2**
- Optional.
  - *Table and Column:* ADDR2
  - *Type:* String

**address3**
- Optional.
  - *Table and Column:* ADDR3
  - *Type:* String
address4
Optional.
Table and Column: ADDR4
Type: String

address5
Optional.
Table and Column: ADDR5
Type: String

address6
Optional.
Table and Column: ADDR6
Type: String

attentionName
Optional.
Table and Column: ATTENTION_NAME
Type: String

attentionPhone
Optional.
Table and Column: ATTENTION_PHONE
Type: String

contactName
Optional.
Table and Column: CONTACT_NAME
Type: String

contactPhone
Optional.
Table and Column: CONTACT_PHONE
Type: String
WIP Class

Use the WIP class XOG object to view inbound and outbound WIP class instances.

Schema Name

nikuxog_wipclass.xsd

Read and Write XML Files

The following XML files are included:
- wipClass_read.xml. Use this file to export WIP class instances from CA Clarity.
- wipClass_write.xml. Use this file to import WIP class instances that were previously exported from CA Clarity.

Prerequisites

None.

Read Filters

The following explicit read filters are used:
- wipclass
  Defines the WIP class name.
- description
  Defines the description for the WIP class.
- shortdesc
  Defines the short description for the WIP class.

Error Handling

The following errors can be thrown:
- Wipclass or description or short description is out of bound.

Schema Mapping

Mappings for the following schema tag name is provided:
- WIP Class (see page 437)
WIP Class (wipclass) Schema Tag

The WIP class tag is part schema mapping for the WIP class XOG object. It has the following attributes:

wipclass
   Required. Defines the unique WIP class name.
   
   **Table and Column:** WIPCLASS
   **Type:** String

description
   Required. Defines the description of the WIP class.
   
   **Table and Column:** DESCRIPTION
   **Type:** String

shortdesc
   Required. Defines the summary of the WIP class description.
   
   **Table and Column:** SHORTDESC
   **Type:** String
Appendix B: Content Object Reference

This section contains the following topics:

- About Content Objects (see page 439)
- Business Alignment (see page 440)
- Corporate Objectives (see page 442)
- Documents (see page 444)
- Status Updates (see page 446)

About Content Objects

CA Clarity provides a collection of portlets, reports, queries, and pages designed to enhance project portfolio management. This content is available out of the box with CA Clarity. The content includes a set of sample data designed for test environments. This appendix provides information on the objects for the content included with CA Clarity.

For more information, see the Project Management User Guide.
Business Alignment

Use the CustomObjectInstances base XOG object to view inbound and outbound Business Alignment instances. Business Alignment instances are created for existing projects and proposals. The CustomObjectInstances service is an entry point to enable XOG communication with instances of custom objects. Instances represent data held within custom objects, not the definition of the objects.

Schema Name

The following schema files are part of this XOG object:

- A CustomObjectInstances read request requires the namespace nikuxog_read.xsd and then the <CustomObjectInstanceQuery> element.
- The write CustomObjectInstances request services are defined by the nikuxog_customObjectInstance.xsd schema.

Read and Write XML Files

The following XML files are included:

- custom_object_instances_read.xml. Use this file to export business alignment instances from CA Clarity.
- custom_object_instances_write.xml. Use this file to import business alignment instances that were previously exported from CA Clarity.

Prerequisites

The referred instances should exist in CA Clarity before using this XOG.

Business Rules and Processing

The following business rules and processing apply to this XOG object:

- Project or proposal parent instance must exist.
- Existing business alignment instances are not deleted.

Read Filters

The CustomObjectInstanceQuery element allows you to filter on instances of one or more custom objects using the following filter attributes:

objectCode
- Refers to the custom object ID as defined in Studio.

instanceCode
- Refers to the custom object instance ID as defined in Studio.
Error Handling

The following errors can be thrown:

- Project does not exist in the system - Invalid parent object instance project code for object project while importing instance instance code for object cbi_biz_alignment.
- Invalid Lookup value
Corporate Objectives

Use the CustomObjectInstances base XOG object to view inbound and outbound Corporate Objectives instances. The CustomObjectInstances service is an entry point to enable XOG communication with instances of custom objects. Instances represent data held within custom objects, not the definition of the objects.

Schema Name

The following schema files are part of this XOG object:

- A CustomObjectInstances read request requires the namespace nikuxog_read.xsd and then the <CustomObjectInstanceQuery> element.
- The write CustomObjectInstances request services are defined by the nikuxog_customObjectInstance.xsd schema.

Read and Write XML Files

The following XML files are included:

- custom_object_instances_read.xml. Use this file to export corporate objectives instances from CA Clarity.
- custom_object_instances_write.xml. Use this file to import corporate objectives instances that were previously exported from CA Clarity.

Prerequisites

None.

Business Rules and Processing

The following business rules and processing apply to this XOG object:

- Existing corporate objectives instances are not deleted.

Read Filters

The CustomObjectInstanceQuery element allows you to filter on instances of one or more custom objects using the following filter attributes:

objectCode
- Refers to the custom object ID as defined in Studio. The object id is cbi_corp_objectives.

instanceCode
- Refers to the custom object instance ID as defined in Studio.
Error Handling

The following errors can be thrown:

- Invalid Lookup value
Documents

Use the CustomObjectInstances base XOG object to view inbound and outbound documents instances. Documents instances are created for existing projects and proposals. The CustomObjectInstances service is an entry point to enable XOG communication with instances of custom objects. Instances represent data held within custom objects, not the definition of the objects.

Schema Name

The following schema files are part of this XOG object:

- A CustomObjectInstances read request requires the namespace nikuxog_read.xsd and then the <CustomObjectInstanceQuery> element.
- The write CustomObjectInstances request services are defined by the nikuxog_customObjectInstance.xsd schema.

Read and Write XML Files

The following XML files are included:

- custom_object_instances_read.xml. Use this file to export documents instances from CA Clarity.
- custom_object_instances_write.xml. Use this file to import documents instances previously exported from CA Clarity.

Prerequisites

The referred instances should exist in CA Clarity before using this XOG.

Business Rules and Processing

The following business rules and processing apply to this XOG object:

- Project or proposal parent instance must exist.
- Existing document instances are not deleted.

Read Filters

The CustomObjectInstanceQuery element allows you to filter on instances of one or more custom objects using the following filter attributes:

objectCode
    Refers to the custom object ID as defined in Studio.

instanceCode
    Refers to the custom object instance ID as defined in Studio.
Error Handling

The following errors can be thrown:

- Project does not exist in the system - Invalid parent object instance project code for object project while importing instance instance code for object cbi_documents.
- Invalid Lookup value
**Status Updates**

Use the CustomObjectInstances base XOG object to view inbound and outbound Status Update instances. Status Updates instances are created for existing projects and proposals. The CustomObjectInstances service is an entry point to enable XOG communication with instances of custom objects. Instances represent data held within custom objects, not the definition of the objects.

**Schema Name**

The following schema files are part of this XOG object:

- A CustomObjectInstances read request requires the namespace nikuxog_read.xsd and then the `<CustomObjectInstanceQuery>` element.
- The write CustomObjectInstances request services are defined by the nikuxog_customObjectInstance.xsd schema.

**Read and Write XML Files**

The following XML files are included:

- `custom_object_instances_read.xml`. Use this file to export status updates instances from CA Clarity.
- `custom_object_instances_write.xml`. Use this file to import status updates instances that were previously exported from CA Clarity.

**Prerequisites**

The referred instances should exist in CA Clarity before using this XOG.

**Business Rules and Processing**

The following business rules and processing apply to this XOG object:

- Project or proposal parent instance must exist.
- Existing status updates instances are not deleted.

**Read Filters**

The CustomObjectInstanceQuery element allows you to filter on instances of one or more custom objects using the following filter attributes:

- **objectCode**
  
  Refers to the custom object ID as defined in Studio.

- **instanceCode**
  
  Refers to the custom object instance ID as defined in Studio.
Error Handling

The following errors can be thrown:

- Project does not exist in the system - Invalid parent object instance project code for object project while importing instance instance code for object cби_status_report.
- Invalid Lookup value
Chapter 7: XOG WSDL

Appendix C: GEL Tag Library Reference

This section contains the following topics:

Tag Libraries (see page 449)
GEL Tag Library (see page 449)
Core Tag Library (see page 470)
SOAP Tag Library (see page 477)

Tag Libraries

Every GEL tag is associated with one of the following tag libraries:

■ GEL Tag Library
  This is a collection of general-purpose, frequently used tags for XML manipulation, variable and expression handling, logging, and this product's database JDBC datasource.

■ Core Tag Library
  The Core tag library contains basic scripting tags.

■ SOAP Tag Library
  This library defines tags that invoke SOAP-based web services and stores the results in GEL variables for subsequent processing.

GEL Tag Library

To use the GEL tag library, include the following namespace declaration in your script.

<gel:script
xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">
**gel:script - Defining GEL Scripts**

This is the root element for all GEL scripts.

This element is the core:jelly:

- **escapeText**
  - Values:
    - true. The tag body is escaped (interpreted as text).
    - false. The body is interpreted as XML.
  - **Default:** true
  - **Type:** Boolean

- **trim**
  - Values:
    - true. The white space inside this tag is trimmed.
    - false. The white space is not trimmed.
  - **Default:** true
  - **Type:** Boolean
**gel:parse - Parsing XML**

Use `gel:parse` to generate an XML document in memory from a file, `InputStream` (obtained with `ftp:get` tag), or GEL script content.

Using other `get` tags, you can:
- Save the output.
- Generate an XML document from an `InputStream`.
- Generate an XML document from GEL script content.

This tag has the following attributes:

- **file**
  - Optional. The file to read. Specify the input path and file name or the `InputStream` from the `ftp:get` tag.
  - If this attribute is not set, the content of this tag is used.
  - **Type:** File or `InputStream`

- **var**
  - Required. The name of the variable that contains the XML document to be generated.
  - **Type:** String

**Example 1**

```xml
<gel:script
   xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">
  <gel:parse var="xmlfdoc" file="e:\temp\BB1.xml"/>
</gel:script>
```

**Example 2**

```xml
<gel:script xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">
  <gel:parse var="xmlfdoc">
    <groups>
      <group code="CTU">CTU Team</group>
      <group code="DS23">SWAT Team</group>
    </groups>
  </gel:parse>
</gel:script>
```
gel:set - Setting XML Document Values

Once you use gel:parse or soap:invoke and have an XML node or document, you can use gel:set to retrieve certain element content or attributes and set the value to a variable. You can also use gel:set to change content (including text and attributes) or add an element with its full structure as a child into another element.

Example

```xml
<gel:script
 xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">

  <!-- point A -->
  <gel:parse var="groups">
    <groups>
      <group code="DS23">SWAT Team</group>
    </groups>
  </gel:parse>

  <!-- point B -->
  <gel:set select="$groups/groups/group" var="groupNode"/>

  <!-- point C -->
  <gel:set select="$groupNode/@code" var="code" asString="true"/>

  <!-- point D -->
  <gel:set value="$\{groupNode\}" select="$groups/groups" insert="true"/>

  <!-- point E -->
  <gel:set value="CTU Team" select="$groupNode/text()"/>

  <!-- point F -->
  <gel:set value="CTU" select="$groupNode/@code"/>

  <!-- point G -->
  <gel:set select="$groups/groups" var="x" asString="true"/>

  <gel:out>${x}</gel:out>

</gel:script>
```

The GEL context contains these values:

- Point A. An XML document, referred by groups, with the specified content. The root element is <groups> and has one <group> sub-element.
- Point B. The document groups is not changed. An XML node, groupNode, is the first <group> element in the document groups.
- Point C. A variable code is created with the content of the code attribute from the node groupNode (that is, DS23).
- Point D. A new XML node with the content specified in the groupNode node is added to the <groups> element of groups. The groups now refers to an XML document whose root element is <groups> that has two sub-elements with the same content. The groupNode still refers to the first <group> element.

Example:

Groups is the whole XML document, and groupNode is the element:

```xml
<groups>
  <group code="DS23">SWAT Team</group>
  <group code="DS23">SWAT Team</group>
</groups>
```

- Point E. The text content of groupNode is changed to CTU Team because groupNode is an element within groups. Document is also changed.

- Point F. The code attribute of groupNode is changed to CTU.

- Point G. The entire XML structure is captured as text to the variable x.

When you print the document referred by groups, you will see:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<groups>
  <group code="CTU">CTU Team</group>
  <group code="DS23">SWAT Team</group>
</groups>
```

**Retrieve XML Document Values**

Use the attributes `var` and `select` to retrieve values from an XML document.

If the select refers to a non-existing path, no value setting is performed (that is, if `var` refers to a variable that is not set in another place, it will be null).

**To retrieve the text content of a node**

```xml
<set var="..." select="$doc/.../node_name/text()"
     asString="true"/>.
```

**To retrieve a certain attribute**

```xml
<set var="..." select="$doc/.../node_name/@attribute_name"
     asString="true"/>
```

**To retrieve a node, including its sub-nodes**

```xml
<set var="..." select="$doc/.../node_name"/>
```
Modify XML Document Values

Use the attributes var and select together to set values in an XML document. The select attribute must refer an existing path. If select = "$doc/group" and there is not an element called group in the document or node referred by doc, an exception will be thrown. If select="$doc/group/text()" and the <group> element does not contain text content, an exception will be thrown.

If the node does not have text content, to set the text content of a node

```
<set value="..." select="$doc/.../node_name"/>
-or-
<set value="..." select="$doc/.../node_name/text()"/>
```

If you reverse the previous two examples, you will get an exception (because a node does not have any child text but you referred to it with text()), or the item you tried to set will be appended to the previous text content.

If you are not sure if the node for which you want to set text content has text content already, it is best to retrieve its text value first and then use core:if to check if it exists before proceeding.

To set the attribute value of node

```
<set value="..." select="$doc/.../node_name/@attribute_name"/>.
```

To set a node into another document

```
<set value="${node_var}" select="$doc/.../node_name"/>
```

You can use attribute insert if you are adding a node to a path or to have this node replace whatever is referred to by the path.

The following describes the gel:set:

**var**

Either var or value is required. The variable to export for the item being iterated over. The variable can be a string, a number, etc.

**Type:** String

**value**

Either var or value is required. If the value is a node, it is inserted into the position specified by select; otherwise the string value is set as the text content or attribute specified by select.

**Type:** Object

**select**

Required. The XPath expression to use to retrieve a value.
Type: org.jaxen.XPath

asString
Optional. If set to "true", the value specified by select is converted to a string and saved into the variable referred to by var. If set to "false", the node specified by select is set to the variable referred by var.

Default: false.
this is ignored when var is not set.

Type: Boolean

insert
Optional. If set to "true", the node referred to by value is inserted as a child node to the node specified by select. If set to "false", the node referred to by value is used to replace the node specified by select.

Default: false. This is ignored when value is not set, or set but not with a node value.

Type: Boolean


**gel:expr - Evaluating Expressions**

Use this tag to evaluate an expression as text. Most often the expression resolves to an XML element as illustrated in the following examples.

**Example 1**

```xml
<gel:script
  xmlns:core="jelly:core"
  xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">
  <gel:parse var="group">
    <group code="CTU">CTU Team</group>
  </gel:parse>

  <core:comment>
    The code is <gel:expr select="$group//@code"/>
  </core:comment>
</gel:script>
```

**Example 2**

The previous example is equivalent to the following gel:set example:

```xml
<gel:script
  xmlns:core="jelly:core"
  xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">
  <gel:parse var="group">
    <group code="CTU">CTU Team</group>
  </gel:parse>

  <gel:set var="code" select="$group//@code" asString="true"/>

  <core:comment>
    The code is ${code}
  </core:comment>
</gel:script>
```

The following describes gel:expr

**select**

Required. The XPath expression to retrieve the value.

**Type:** XPath
**gel:parameter - Defining Parameters**

Use this tag to define parameters that can be used in a GEL script.

**Example**

```xml
<gel:script
 xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">

 <gel:parameter var="hostname" default="http://localhost/niku/xog"/>
 <gel:parameter var="username" default="admin"/>
 <gel:parameter var="password" default="niku2000" secure="true"/>

 <gel:out>Host = ${hostname}</gel:out>
 <gel:out>User = ${username}</gel:out>

</gel:script>
```
Use gel:parameter Instead of core:set

When a GEL script is executed from the console, there is no difference between using gel:parameter and core:set.

When gel:parameter is executed as a process, all parameters that were defined using the <gel:parameter> tag appear with input boxes on the action definition page. You can enter a value for a parameter to override the default value in the script.

You should use gel:parameter for values that may be changed by process administrators (such as URL, hostname, username, etc.). Also use this for values which should be kept discrete, like passwords.

You can only define one parameter name at a time. For example, if you use logic such as "if a certain condition, log in as userA, otherwise userB," instead of defining "username" in two places, use this parameter to log in, define two properties "usernameA" and "usernameB", and then use the <core:set> tag to pick one of those two properties to set into a variable in the "if" block.

A parameter can be used later just like other variables (that is, ${var}).

The following describes gel:parameter:

**var**

Required. The parameter name.

*Type:* String

**default**

Optional. The parameter default value. Provide this value if you want the script to be executable from the console (even if this parameter is not secure).

*Type:* Object

**secure**

Required. Set this attribute to "true" if the parameter content should not be shown in plain text to process administrators.

*Default:* false

*Type:* Boolean
gel:getDocument - Requesting XML Documents

A process can be invoked in the following ways:

- Manually
- Invoked by a job scheduler
- Invoked by a request on the web service around the process engine.

When a process is invoked as a web service, the request is an XML document. You can use this tag to get that document, find what needs to be done, and then perform actions accordingly.

If an XML document is set using the gel:setDocument tag in one step of a process, you can use this tag to retrieve the document a later step of the same process.

var

Required. The name of an XML document variable which was set in the previous step of the same process. If this step is the first step, this variable is the body of the SOAP request that is sent to the process engine web service.

Type: String

gel:setDocument - Passing XML Documents

Use this tag to pass an XML document that was generated in one step of a process to the next step. This allows you to write the processing logic in separate steps.

For example, you can invoke one web service, save the response in a step, then retrieve it and use it to invoke XOG in another step.

var

Required. The name of an XML document variable which is to be passed to the next step in the same process.

Type: String
**gel:persist - Persisting Variables**

When you set a variable in a GEL script, you can only use it when executing that script. Sometimes when the GEL script is executed in a process engine, you need to share a value in other scopes such as:

- Globally among all process GEL scripts,
- Only with GEL scripts in a certain process within all process executions (such as a single variable that stores the time a *Remedy Clarity Incident Sync* process was most recently launched),
- Only with GEL scripts in different steps of a process during one process execution,
- Use `<gel:persist>` to achieve variable value sharing. Once a value is persisted, you can use it directly in proper scripts without defining any special tags. For example, after `<gel:persist var="hostname" value="localhost" scope="INSTANCE"/>` is processed, you can refer to `${hostname}` directly in any GEL script of this process during this process execution.

You can access a persisted value with a PROCESS scope using scripts from that process (even if the process, its steps, and GEL scripts change). If a process is deleted and then recreated, it is considered to be a new process and all values persisted before with PROCESS scope are not available to the new process (even if the new process has the same process name, code, or steps as the deleted one).

**var**

Required. The variable to be persisted.

*Type*: String

**value**

Optional. The value of the variable. It has to be a string (formatted date strings are acceptable). When this attribute is not set, the tag content is used as the value to be persisted. If the value being persisted contains special characters, such as a new line, do not use this attribute, use the text content instead.

*Maximum Length*: 4000 characters. Longer strings are truncated.

*Type*: String

**scope**

Required. Specifies the scope of the variable.

*Values:*

- GLOBAL. Set once, use it anywhere.
- PROCESS. Set once, use it anywhere in the same process.
- INSTANCE. Set once, use it anywhere in the same process during the current execution.

*Type*: String
Example

The following example persists the Clarity built-in variables \texttt{gel\_objectInstanceId} and \texttt{gel\_processInstanceId} throughout this process instance as \texttt{myObjectId} and \texttt{myProcessId}.

\begin{verbatim}
<gel:script
    xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">

    <gel:persist var="myObjectId" value="${gel_objectInstanceId}"
        scope="INSTANCE"/>
    <gel:persist var="myProcessId" value="${gel_processInstanceId}"
        scope="INSTANCE"/>

</gel:script>
\end{verbatim}
**gel:notify - Sending Notifications**

Use this tag to send email. The email content is the text content of this tag, followed by process messages logged thus far during the current process.

Email server information is derived from the properties.xml file of the installation.

**from**

Required. The sender's email address.

*Type: String*

**fromName**

Optional. The name of the sender.

*Type: String*

**to**

Required. The recipients' email addresses (delimited by commas, semicolons, or spaces).

*Type: String*

**subject**

The email subject.

*Type: String*

**level**

Optional. Set this to:

- WARNING to have the email sent only if there are warning or error messages.
- ERROR to have email sent only if there are error messages. Only error messages are included in the message.

If this attribute is not specified, email is sent no matter how many log messages are retrieved. All process messages logged thus far are included.

*Type: String*

**Example**

This example sends a notification if an error had been logged with `<gel:log>`.

```xml
<gel:script
xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">

<gel:notify from="username@mailserver.com"
fromName="Clarity Admin"
to="user@somedomain "
subject="There was a process error"
level="ERROR">
```
A process error was received.
</gel:notify>

</gel:script>

gel:email - Sending Email Messages

Use this tag to send an email. The email content is the text content of this tag. Email server information is derived from the properties.xml of the installation.

from

Required. The sender's email address.

Type: String

fromName

Optional. The sender's name.

Type: String

to

Required. The recipients' email addresses (delimited by commas, semicolons, or spaces).

Type: String

subject

Required. The email subject.

Type: String

Example

This example sends a simple email:
<gel:script
xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">

<gel:email from="username@mailserver.com"
fromName="Clarity Admin"
to="user@somedomain"
subject="Simple email">
Hello World.
</gel:email>

</gel:script>
**gel:formatDate - Formatting Time Strings**

This tag provides a formatted time string which one can used as a part of a file name, appended to a comment line, or inserted into a database. The following example:

```xml
<gel:out>Hello World! Now it is <gel:formatDate format="h 'o''clock' a, zzzz, d MMM yyyy"/></gel:out>
```

generates the following output:
Hello World! Now it is 4 o'clock PM, Pacific Standard Time, 24 Mar 2005.

This tag has the following attributes:

- **format**
  
  Optional. Specifies how time displays in java.text.SimpleDateFormat format.
  
  **Note:** Go to [http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html](http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html)
  
  **Default:** yyyy-MM-dd HH:mm:ss
  
  **Type:** String

- **stringVar**
  
  Optional. This variable refers a formatted date string. If this attribute is not set, the formatted string is used in the content of this tag's parent element.
  
  **Type:** String

- **dateVar**
  
  Optional. The variable, of type java.util.Date, referred to by this name is formatted as a string. If this attribute is not set, the current time is used.
  
  **Type:** String

**Example 1**

This example formats the current date and time into the format the XOG requires for investment start/finish dates.

```xml
<gel:script
 xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">

<gel:out>
  <gel:formatDate format="yyyy-MM-dd'T'HH:mm:ss"/>
</gel:out>

</gel:script>
```

**Example 2**
This example formats the specified date and time into the format the XOG requires for investment start/finish dates. Notice the use of the Java class java.util.Date and the <core:new>, <core:invoke> and <core:arg> tags.

```xml
<gel:script
   xmlns:core="jelly:core"
   xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">

   <core:new className="java.util.Date" var="date"/>
   <core:invoke on="${date}" method="parse">
      <core:arg value="2009/03/27"/>
   </core:invoke>

   <gel:out>
      <gel:formatDate format="yyyy-MM-dd'T'HH:mm:ss" dateVar="date"/>
   </gel:out>

</gel:script>
```
**gel:parseDate - Parsing Time Strings**

This tag takes a formatted string, then generates a date instance.

This tag uses the following attributes:

**format**

Optional. Indicates how the string is formatted in `java.text.SimpleDateFormat` format.

**Note:** Go to [http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html](http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html)

**Default:** `yyyy-MM-dd HH:mm:ss`

**Type:** String

**stringVar**

Optional. This variable refers to the string to be parsed. If the string does not have the format specified by the format attribute, a parsing exception is thrown.

If this attribute is not set, the text content of this tag is used as the string.

**Type:** String

**dateVar**

Required. The parsed date is stored as a `java.util.Date` and referred to by this variable name.

**Type:** String

**Example**

This example parses a date from a string, then formats that date using `<gel:formatDate>`.

```xml
<gel:script
    xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">

    <gel:parseDate dateVar="date" format="yyyy-MM-dd">2009-03-27</gel:parseDate>

    <gel:out>
        My date was: <gel:formatDate format="yyyy-MM-dd'T'HH:mm:ss" dateVar="date"/>
    </gel:out>

</gel:script>
```
**gel:setDataSource - Specifying Data Sources**

Use this tag to identify the CA Clarity PPM database.

```xml
<gel:setDataSource dbId="niku"/>
```

When you access the CA Clarity PPM database, you only need to know its database ID (that is, you do not need to provide other access information such as username).

This tag uses the following attribute:

**dbId**

- **Required.** The database ID.
- **Type:** String

**gel:nsqlQuery - Executing NSQL Queries**

This tag allows you execute an existing NSQL query, or define a new ad-hoc query on the fly to retrieve data from the database, storing the results to the specified variable.

**Examples**

```xml
<gel:setDataSource dbId="niku" var="dataSource"/>

<gel:nsqlQuery queryId="usercountbylicensetype" var="resultSet">
  <gel:nsqlParameter name="license_wildcard" value="*"/>
</gel:nsqlQuery>
<core:forEach items="${resultSet}" var="row">
  <gel:out>Row Contents: '${row}'.</gel:out>
</core:forEach>

<gel:nsqlQuery var="resultSet">
  <![CDATA[
    SELECT   @SELECT:U.USER_NAME:USER_NAME@,
            @SELECT:U.ID:USER_ID@
    FROM     CMN_SEC_USERS U
    WHERE    @FILTER@ ]]> 
  <gel:nsqlParameter name="user_name_wildcard" value="admin*"/>
</gel:nsqlQuery>
<core:forEach items="${resultSet}" var="row">
  <gel:out>Row Contents: '${row}'.</gel:out>
</core:forEach>
```
**gel:log - Logging Messages**

Use this tag to insert status messages into the process engine log table.

```xml
<gel:script
    xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">

    <gel:log level="warn" category="Employee Data"
        message="No record returned."/>

</gel:script>
```

This tag logs messages as a process message in the BPM_ERRORS table when this script runs as a custom step in a process. If the process is running from the console, the message is inserted into the standard log file.

**message**

Optional. The message to log. The message can be set as a value attribute or as the content of this tag.

Type: String

**category**

Optional. Use this to distinguish logs. It can be concatenated from business data type, file name, developer ID, etc.

Type: String

**level**

Optional. This is the warning level. Choose from the following:

- DEBUG
- ERROR
- FATAL
- INFO
- WARN

This attribute is not case sensitive. For example, WARN, warn, and Warn are the same.

A process message has only three levels: INFO, WARNING, and ERROR, while a logger message in the log file can have all levels. When a message is logged as a process message, DEBUG and INFO messages are logged as INFO messages, WARN messages are logged as WARNING messages, and ERROR and FATAL messages are logged as ERROR messages.

Default: INFO

Type: Level

```
var
```
Optional. A variable name into which the log message should be stored. Use this when you want to save log messages for other purposes such as sending emails.

If the variable is:

- Not set. A StringBuffer is created for storing the message; it can later be referred to using this variable name.
- Already a StringBuffer. The StringBuffer will be appended to the log message.
- A string. A StringBuffer is created for storing the string referred to by this variable followed by the log message; it can later be referred to using this variable name.

**Type:** String

### gel:out - Printing to the Console

This tag prints the content of this tag to the system console. It does not have any attributes.

Use this tag only when you are using the console to debug and the GEL script is not running as a process. For example,

```
<core:set var="x" value="file.rows[2][3]"/>
<gel:out>${x}</gel:out>
```

If you have a variable that contains an XML Node, including an XML document and you want to print it, combine gel:out with gel:expr:

```
<gel:parse var="doc">
  <groups>...</groups>
</gel:parse>
<gel:out><gel:expr select="$doc/groups"/></gel:out>
```
The tags in this section are a useful subset of the jelly:core tag library. Go to http://jakarta.apache.org/commons/jelly/tags.html for Jakarta Jellytag descriptions.

The following additional tags can invoke Java class methods directly:

- core:new
- core:invoke
- core:invokeStatic

The following tags are also useful for controlling flow in your script:

- core:if
- core:switch, core:case, core:default
- core:choose, core:when, core:otherwise
- core:forEach
- core:while
- core:break

Include the following namespace declaration in your script to use this tag library:

```xml
<gel:script xmlns:core="jelly:core"...>
```
**core:catch - Catching Exceptions**

Use the Jelly fault-handling tags to catch exceptions and exit gracefully when a process failure occurs. Use the `<j:catch>` tag to capture exceptions in the `ex` variable. Outside of the catch tags, you can check the `ex` variable and write it to the console using `gel:out`.

**Example**

```gel:script
<gel:script
 xmlns:core="jelly:core"
 xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">
 <!-- this gel:set will throw an exception -->
 <core:catch var="exception">
   <gel:set select="$bad/text()" var="mynode"/>
 </core:catch>

 <core:if test="${exception != null}">
   <gel:out>Caught Exception was: ${exception}</gel:out>
 </core:if>

</gel:script>
```

**core:set - Setting Variables**

This sets a variable from the result of an expression.

**defaultValue**

Sets the default value to use if the value expression results in a null value or blank string.

**Type:** org.apache.commons.jelly.expression.Expression

**encode**

When set to:

"1", the body of the tag is encoded as XML text. When "<" and ">" are encountered in the tag body, they are encoded as "&lt;" and "&gt;".

"0", the body is not encoded.

Use this only if this tag is specified with no value so that the text body of this tag can be used as the body.

**Type:** Boolean
escapeText
When set to:
“1”, the body of the tag is escaped (interpreted as text).
“0”, the body is interpreted as XML.
**Default:** “1” (text)
**Type:** Boolean

property
Indicates the property name to set on the target object.
**Type:** java.lang.String

scope
Sets the scope of this variable. For example when set to "parent", this value is in the parent scope. When Jelly is run from inside a servlet then other scopes are available such as "request", "session", or "application".
Other applications may implement their own scopes.
**Type:** java.lang.String

target
Sets the target object on which to set a property.
**Type:** java.lang.Object

trim
When set to:
“1”, whitespace inside this tag is trimmed.
“0”, whitespace is not trimmed.
**Default:** “1” (trimmed).
**Type:** Boolean

value
Sets the expression to evaluate.
**Type:** org.apache.commons.jelly.expression.Expression
**var**

Sets the variable name to define for this expression.

**Type:** java.lang.String

**Example**

This example shows setting strings and numbers.

```xml
<gel:script
 xmlns:core="jelly:core"
 xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">

  <core:set var="color" value="blue"/>
  <gel:out>Color is ${color}</gel:out>

  <core:set var="age" value="39"/>
  <gel:out>My age is ${age - 18}</gel:out>

</gel:script>
```
**core:forEach - Iterating over Elements**

This iterates over elements. It has the following attributes:

**begin**
- Sets the starting index value (for first element in the array).
- **Type:** int

**end**
- Sets the last index value.
- **Type:** int

**escapeText**
- When set to:
  - 1 - The body of the tag is escaped (interpreted as text).
  - 0 - The body is interpreted as XML.
- **Default:** 1
- **Type:** Boolean

**indexVar**
- Sets the variable into which the current index counter is exported.
- **Type:** java.lang.String

**items**
- Sets the expression used to iterate over. This expression may resolve to an iteration, collection, map, array, enumeration, or comma-delimited string.
- **Type:** org.apache.commons.jelly.expression.Expression

**step**
- Sets the index increment step.
- **Type:** int

**trim**
- **Values:**
  - 1. The whitespace inside this tag is trimmed.
  - 0. The whitespace is not trimmed.
- **Default:** 1
- **Type:** Boolean
var

Sets the variable into which the item being iterated over.

Type: java.lang.String

varStatus

Sets the variable into which the current status is exported. The status is an implementation of the JSTL LoopTagStatus interface that provides the following bean properties:

- current. The current value of the loop items being iterated.
- index. The current index of the items being iterated.
- first. If true, this is the first iteration.
- last. If true, this is the last iteration.
- begin. This indicates the starting index of the loop.
- step. This indicates the number by which the loop is iterated, for example:
  - 1. Indicates each loop increments by one index (for example: 1, 2, 3).
  - 2 indicates each loop increments by two index numbers (for example: 1, 3, 5).
- end. Indicates the last index in the loop.

Type: java.lang.String

Example

This example iterates through the properties in the file test.properties that ships with the XOG client and prints out each property.

```gel:script
<gel:script
  xmlns:core="jelly:core"
  xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">
<!- print out each value we find -->
<core:forEach items="A, B, C, 1, 2, 3" var="value">
  <gel:out>Value = ${value}</gel:out>
</core:forEach>
</gel:script>```
core:if - Evaluating Conditionally

This tag evaluates the body based on some condition.

This tag has the following attributes:

**escapeText**

When set to:
- **1.** The body of the tag is escaped (interpreted as text).
- **0.** The body is interpreted as XML.

**Default:** 1

**Type:** Boolean

**trim**

When set to:
- **1.** The whitespace inside this tag is trimmed.
- **0.** The whitespace is not trimmed.

**Default:** 1.

**Type:** Boolean

**test**

Sets the Jelly expression to evaluate. If this returns true, the body of the tag is evaluated.

**Type:** org.apache.commons.jelly.expression.Expression

**Example 1**

This example tests the value of a variable in a `<core:if>` statement.

```gel
<gel:script
 xmlns:core="jelly:core"
 xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">
  <core:set var="color" value="blue"/>
  <core:if test="${color == 'blue'}">
    <gel:out>Color matched blue!</gel:out>
  </core:if>
</gel:script>
```

**Example 2**
This example tests the numeric value. Notice that the > symbol has been escaped in the XML as > is a reserved character.

```xml
<gel:script
   xmlns:core="jelly:core"
   xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">

   <core:set var="age" value="10"/>

   <core:if test="${age &gt; 5}">
     <gel:out>Age is greater than 5</gel:out>
   </core:if>

</gel:script>
```

**SOAP Tag Library**

Use the XML SOAP tags in this section to invoke a SOAP-based external or internal web service such as the XOG API.

Include the following namespace declaration in your script to use this tag library:

```xml
<gel:script
   xmlns:gel="jelly:com.niku.union.gel.SOAPTagLibrary">
```
**soap:invoke - Invoking SOAP Web Services**

Use this tag to invoke a web service at a specified endpoint and assign a name to the resulting XML document. You can use subsequent tags to access the tag's variables and extract data from the document. The invoke tag can contain sub-tags, including soap:envelope and soap:attachment.

This tag has the following attributes:

**endpoint**
Required. Specifies either the keyword 'internal' or the URL of the web service to be invoked.

**Values:**
- 'internal'. CA Clarity endpoint URL is used automatically.
- Fully qualified URL.

**Type:** String

**var**
Optional. Contains the response from the web service. The response is of type org.w3c.dom.Document.

**Type:** String

This tag has the following subtags:
- soap:message
- soap:attachment

**Example**

```xml
<soap:invoke endpoint="internal" var="result">
  <soap:message>...</soap:message>
</soap:invoke>
```

```xml
<soap:invoke endpoint="${serviceUrl}" var="result">
  <soap:message>...</soap:message>
</soap:invoke>
```

**soap:envelope - Generating a SOAP Envelope**

This tag generates a SOAP envelope which can be used by soap:invoke to send a SOAP request. It includes the following header and body tags:
- soap:header
- soap:body
soap:header - Specifying the SOAP Header

This tag contains the SOAP header, which should be included in a SOAP envelope. You choose which data to include.
soap:body - Specifying the SOAP Body

This tag controls the SOAP body (which should be included in a SOAP envelope). You can control which data to include. You can write content into this tag as illustrated in SOAP Examples, or you can write content as an attribute of this tag as illustrated in the following example.

This tag has the following attribute:

xml

Optional. Sets the source of the soap:body. If this attribute is set, the content of the document variable, which can be set by gel:parse or ftp:get, is used as the content of this soap:body tag (and the body of this tag is ignored). If there is an XML file you want to use as the content of SOAP body, use gel:parse to read the file and set this attribute.

Type: org.w3c.dom.Document

Example

This example executes an NSQL query through the XOG web service and writes the results to a tab-delimited file.

```xml
<gel:script xmlns:core="jelly:core"
xmlns:xog="http://www.niku.com/xog"
xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary"
xmlns:soap="jelly:com.niku.union.gel.SOAPTagLibrary"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
xmlns:f="jelly:com.niku.union.gel.FileTagLibrary"
xmlns:nikuq="http://www.niku.com/xog/Query"

<!-- Construct the Query API request for the NSQL query -->
<gel:parse var="xoginput">
  <Query xmlns="http://www.niku.com/xog/Query">
    <Code>cats.resourceProfile</Code>
  </Query>
</gel:parse>

<soap:invoke endpoint="internal" var="xogresponse">
  <soap:message>
    <soapenv:Envelope>
      <soapenv:Header>
        <Auth>
          <Username>admin</Username>
          <Password>niku2000</Password>
        </Auth>
      </soapenv:Header>
      <soapenv:Body>
        <!-- Insert XML content here -->
      </soapenv:Body>
    </soapenv:Envelope>
  </soap:message>
</soap:invoke>
```
<gel:include select="$xoginput"/>
</soapenv:Body>
</soapenv:Envelope>
</soap:message>
</soap:invoke>

<!-- Extract the sessionID so we may logout later -->
gel:set asString="true"
    select="$xogresponse/xog:SessionID/text()"
    var="sessionID"/>
gel:out>SessionID = ${sessionID}</gel:out>

<!-- Extract the records -->
gel:set select="$xogresponse/nikuq:QueryResult/nikuq:Records"
    var="records"/>

<!-- Create a tab-delimited file from the results -->
<f:writeFile fileName="projectData.txt"
    delimiter="&amp;#9;" embedded="true">
gel:forEach select="$records/nikuq:Record" var="xog_record">
    <f:line>
        gel:forEach select="$xog_record/*" var="xog_data">
            gel:set var="xog_data" select="$xog_data/text()"
                asString="true"/>
            <f:column value="${xog_data}"/>
        </gel:forEach>
    </f:line>
</gel:forEach>
</f:writeFile>

<!-- Now log out -->
<soap:invoke endpoint="internal" var="logout">
    soap:message
        soapenv:Envelope
            soapenv:Header
                Auth
                    xog:SessionID=${sessionID}</xog:SessionID>
                </Auth>
          </soapenv:Header>
            soapenv:Body>
                xog:Logout/
            </soapenv:Body>
        </soapenv:Envelope>
    </soap:message>
</soap:invoke>

gel:out>Output written to projectData.txt</gel:out>

gel:script</gel:script>
SOAP Tag Library

soap:attachment - Attaching Files to SOAP Requests

This tag specifies the file to be attached in the SOAP request.

This tag has the following attributes:

- **dir**
  - Required. The directory on the local disk where the attachment file is located.
  - **Type:** String

- **fileName**
  - Required. The file to be attached with the SOAP request.
  - **Type:** String

**Example**

```xml
<soap:attachment dir="$dir" fileName="$file"/>
```

soap:message - Specifying SOAP XML Messages

The tag contains the actual SOAP XML message. This includes the SOAP envelope, header and body tags.

**Example**

```xml
<soap:message>
<obj:ReadGroup xmlns:obj="http://www.niku.com/xog/Object">
  <DataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../xsd/xog_read.xsd">
    <xog:Header version="7.5"
      externalSource="NIKU"/>
    <xog:Query>
      <xog:Filter name="code" criteria="OR">
        ProjectManager,PortfolioManager,XOGTestGroup
      </xog:Filter>
    </xog:Query>
  </DataBus>
</obj:ReadGroup>
</soap:message>
```
Example: XOG Login and Read Objects Example

```xml
<gel:script
 xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary"
 xmlns:soap="jelly:com.niku.union.gel.SOAPTagLibrary"
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xog="http://www.niku.com/xog">

<soap:invoke endpoint="internal" var="auth">
    <soap:message>
        <soapenv:Envelope>
            <soapenv:Header>
                <Auth xmlns="http://www.niku.com/xog">
                    <Username>admin</Username>
                    <Password>clarity</Password>
                </Auth>
            </soapenv:Header>
            <soapenv:Body/>
        </soapenv:Envelope>
    </soap:message>
</soap:invoke>

<soap:invoke endpoint="internal" var="result">
    <soap:message>
        <soapenv:Envelope>
            <soapenv:Header>
                <Auth>
                    <xog:SessionID>
                        <gel:expr select="$auth//xog:SessionID/text()"/>
                    </xog:SessionID>
                </Auth>
            </soapenv:Header>
            <soapenv:Body>
                <obj:ReadGroup xmlns:obj="http://www.niku.com/xog/Object">
                    <NikuDataBus
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        xsi:noNamespaceSchemaLocation="../xsd/xog_read.xsd">
                        <Header version="7.5" externalSource="NIKU"/>
                        <Query>
                            <Filter name="code" criteria="OR">
                                ProjectManager,PortfolioManager,XOGTestGroup
                            </Filter>
                        </Query>
                    </NikuDataBus>
                </obj:ReadGroup>
            </soapenv:Body>
        </soapenv:Envelope>
    </soap:message>
</soap:invoke>
```

Chapter 7: XOG WSDL 483
<soap:invoke>
  <soap:invoke endpoint="internal" var="auth">
    <soap:message>
      <soapenv:Envelope>
        <soapenv:Header>
          <Auth>
            <xog:SessionID>
              <gel:expr select="$auth//xog:SessionID/text()"/>
            </xog:SessionID>
          </Auth>
        </soapenv:Header>
        <soapenv:Body>
          <xog:Logout/>
        </soapenv:Body>
      </soapenv:Envelope>
    </soap:message>
  </soap:invoke>
</soap:invoke>
Example: Execute External Web Services with Attachments

The following example places the XOG output as an XML document in the variable result. This example uses a hypothetical external web service "UploadFile."

```xml
<gel:script
 xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary"
 xmlns:soap="jelly:com.niku.union.gel.SOAPTagLibrary"
 xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
>
 <soap:invoke endpoint="${serviceUrl}" var="result">
  <soap:message>
   <soap-env:Envelope>
    <soap-env:Header>
     <AuthId>${authId}</AuthId>
     <Locale>en_US</Locale>
    </soap-env:Header>
    <soap-env:Body>
     <UploadFile xmlns="xxx">
      <NewFile>
       <Name>${file}</Name>
       <ReplaceExisting>true</ReplaceExisting>
      </NewFile>
     </UploadFile>
    </soap-env:Body>
   </soap-env:Envelope>
  </soap:message>
  <soap:attachment dir="${dir}" fileName="${file}"/>
 </soap:invoke>

<gel:out>Out: <gel:expr select="$result"></gel:out>
</gel:script>
```