#

# GEL bits and pieces

A basic guide to getting started with Generic Execution Language (GEL)
a feature of ClarityTM

Why use GEL?
GEL is a scripting facility to execute SQL queries and for:

* read or write to any delimited file including those on local disks, network disks or disk arrays.
* execute SQL to read or update data
* use JDBC to access RDBMS to read or write data
* capture or send data in XOG format.
* read or write to any SOAP-based web service.
* upload or download to FTP servers.
* sending email notifications
* **GEL makes ‘custom scripts’ in Clarity processes!**

Version: 1.0

December 2006

Paul Maxwell

CA

# GEL Setup

To establish GEL on a workstation for command line use you need:

## JRE

Download from the Clarity server and install. Defaults are OK but you may change the location to be directly of the root instead of under “Program Files” (more on that next).

## System variable JAVA\_HOME

You should now establish the system variable JAVA\_HOME and this needs to point to the directory that OWNS the \bin sub-directory. If you install the JRE using defaults this most likely under c:\Program Files. Probably C:\Program Files\Java\j2re1.4.2\_01

Instead of using the string “Program Files” you can usually substitute “Progra~1”

 My Computer > Properties > Advanced > Environment Variables create this system varlable.

To check it, at a command prompt just use ‘set’, JAVA\_HOME should be listed:



## XOG

The GEL command line is actually included in the XOG Client, so download that from Clarity (Admin > Client Downloads). You do not have to use the installer version of XOG, the zip version is normally fine and allows more flexibility if you are likely to be using different versions of Clarity or are testing new fix pack levels.

## Editor

VERY helpful if you have a good XML editor (e.g. XMLSpy)

However this will only assist with “well-formedness” (unlike XOG files) as there is no .xsd specified to validate against.

## Command Prompt Here

A useful (but not essential) utility is to install or create “command prompt here”. Do a Google search if you need it or detail on what it is.

# Command Line GEL

Once you are setup you should now be able to go to the XOG\bin directory in a command window (this is where ‘command prompt here’ helps) at enter the command ‘gel’



You are almost setup now. You can try ‘Hello World’ (see later)

## Properties.xml

If this error displays in command line GEL

C:\XOG\753fp02\bin>gel ../\_gel/my\_first.gel.xml

java.io.FileNotFoundException: C:\XOG\753fp02**\.setup\templates\config\properties**

…..

Internal error: file:/C:/XOG/753fp02/bin/../\_gel/my\_first.gel.xml:28:39: <gel:se

tDataSource> Error unmarshalling config file: properties.xml: C:\XOG\753fp02**\.se**

**tup\templates\config\properties.xml (The system cannot find the path specified)**

do this:

It seems that if you want to do anything significant with GEL you are going to need a copy of your Clarity server’s ‘properties.xml’. This will be on the server running the NSA under $NIKU\_HOME$/config

Now once you have that, you need to create a folder structure starting at the same level as your XOG\bin folder **XOG\.setup\templates\config**

You may need to use the old command line
MD .setup

As windows does not usually like folders starting with a dot.

## Database Connection

When attempting a Clarity database connection, it is the database id from properties.xml that must be specified in GEL. This is NOT displayed in the NSA so unless someone has specifically edited
that file is should always be Niku.

THAT value is used in the following manner:

<gel:setDataSource dbId="**Niku**"/>

# GEL Documentation

## Integration Guide

The major source of GEL documentation is found in the “Integration Guide” (pdf) that ships with Clarity. Please refer to this.

## CA Support

CA’s support for Clarity as of now continues through <http://support.niku.com> but will progressively be migrated to CA’s <http://supportconnect.ca.com>. The Niku knowledgebase (KB) has some information on GEL, as at Dec. 2006 the major recommendations from the KB are contained in this document.

## Further Reading

GEL is based upon the Apache ‘jelly’ project, documentation from there might be helpful:

<http://jakarta.apache.org/commons/jelly/tags.html>

# Hello World

From KB 7767: The Hello World example in the integration guide is wrong, use this instead:

<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>



Actually, the classic Hello World should just be:

<gel:script xmlns:core="jelly:core" xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary">
<gel:out>Hello World!</gel:out>
</gel:script>

# GEL is contained in XML so watch for <> & and others

ALL GEL scripts are actually contained within XML, so all XML rules apply to structure, tags and special characters. That’s why a good XML editor is recommended (see Setup). In addition to these general comments be particularly aware of the Greater Than/Less Than restriction within SQL

Resolution ID: 7743.00000

Subject: Getting error when using Greater then or less then (>, <) in where clause of SQL Query section in GEL Script

Keywords: Gel Script error with greater (>) operator in the Sql:query tag

Product: Clarity
Version: 7.5.2

Description:

Getting error when using Greater then or less then (>, <) in where clause of SQL Query section in GEL Script

Steps to Reproduce:

1. Created the Gel script

2. Added the SQL Query section

Example:

<sql:query var="results">

select count(\*) as projcount from srm\_projects where id > 5000000

</sql:query>

3. run the command to validate the file, it gives error as

Internal error: file:/P:/sql\_test3.xml:16:27: <sql:query> select count(\*

) as projcount from srm\_projects where id &gt; 5015143: [Microsoft][ODBC SQL Ser

ver Driver][SQL Server]Line 1: Incorrect syntax near ';'.

CAUSE:

=======

In Jelly <sql:query> tag the "<" or ">" operators are not allowed. So this is an issue with Jelly and not with Gel Script.

RESOLUTION:

============

**Use "between" in the WHERE clause instead of '<' or '>'.**

# GEL Parameters and Variables

## Automatically provided variables

[From KB 7306] Available in Clarity 7.5.1 FP03 onwards

The following parameters are available to any custom action gel script **that is associated with a process**. All of the built-in parameters have a "gel\_" prefix. All of these parameters are of data type - numeric.

**1.** Object Instance Id: This always -1 if there is no object associated with a process. If there is an object associated with the process then this will return the appropriate object instance id. **${gel\_objectInstanceId}**

**2.** Process Id: The process identifier. **${gel\_processId}** All instances of this process will share this same identifier.

**3.** Process Instance Id: The instance identifier of the process **${gel\_processInstanceId}** - all instances will get a unique value for this parameter.

*Suggestion: make use of these Ids in Error logging.*

## User Specified Variables

To declare a variable of your own use the <core:set..></core:set> tag:

<core:set var="v\_ProjectID">PRJ678910</core:set>

*Suggestion: always use at least ‘v\_’ to start variable names*

*Further suggestion: indicate data type ‘vs\_’ is a string variable,
 ‘vn\_’ is a number variable*

## Parameters

To declare a parameter to your scripts use the <gel:parameter … /> tag:

 <gel:parameter var="p\_sender" default="clarity@ca.com"/>

*Suggestion: always use ‘p\_’ to start parameter names*

Parameters make most sense when the GEL is contained within Clarity process. Once you declare parameters they will be revealed through the Clarity UI like so:



### Using gel:parameter Instead of core:set

<gel:parameter var="hostname" default="http://localhost:80//xog"/>

<gel:parameter var="username" default ="admin"/>

<gel:parameter var="password" **secure="true"**/>

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 Clarity 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 (i.e. ${var}).

## Using Variables and Parameters

Both Variables and Parameters can have their values changed by using the <core:set..></core:set> tag pair.

*Suggestion: you should not change parameter values within a script*

You reference Variable/Parameter values by containing the name within **${** and **}**.

At times you need to include this within double quotes to suit XML syntax, and for strings you may need to use single quotes to suit SQL syntax.

XML syntax needs double quotes at times:
<core:set var="vs\_Holder" value="${gel\_objectInstanceId}"/>

SQL syntax needs single quotes at times:
where text\_field = ‘${vs\_MyVar}’

An exception to the ${var} wrapping occurs when manipulating xml files, see **XXXXXXX**.

## GEL is case sensitive, take extreme care with variables

Information contained within GEL tags is case sensitive. So, for instance, if you declare a variable like this:

<core:set var="v\_**P**roject**ID**">PRJ-123456</core:set>

And then attempt to use that value with an unmatched name, it will cause logic errors in your code:

<gel:out>${v\_projectid}</gel:out>

BUT also be aware, you cannot “catch” these errors!



# Using SQL within GEL

GEL contains the ability to run SQL which therefore gives GEL the ability to access data within Clarity for a whole range of purposes.

## Making a Database Connection

The first step toward using SQL within GEL is to make a connection to the database.

There are several methods for making the connection to Clarity. This is recommended method. Also see ‘’Database Connection’ on page 3.

### <gel.setDataSource/>

The important part is “gel” <gel

Instead of <sql

Details:

Resolution ID: 7733.00000

Subject: Tips on using JDBC in GEL

Keywords: Tips on using JDBC in GEL

Product: Clarity

Version: 7.5.3

Description:

USING JDBC IN GEL

CAUSE:

======

N/A

WORKAROUND

=============

N/A

RESOLUTION/STATUS:

====================

Use the <gel:setDataSource/> tag instead of the <sql:setDataSource/> tag. The GEL version takes in a database id attribute as follows:

<gel:setDataSource dbId="Niku" [var="clarityDS"]/>

**This tag will take the connection properties including the password from Clarity's properties. Niku is the ID of the default internal Clarity database. In NSA you can add as many external database definitions as you like, and also reference them like this. So if you add a second database connection definition in NSA called "ORA-FIN" you can reference those values with:**

<gel:setDataSource dbId="ORA-FIN"/>

Note that the var attribute is optional. If you do not specify a variable, it defaults and subsequent sql:query etc. calls will use that default. In other words, you can do:

<gel:setDataSource dbId="Niku"/>

and

<sql:query var="result">

SELECT 1 from dual

</sql:query>

OR

<gel:setDataSource dbId="Niku" var="clarityDS"/>

and

<sql:query dataSource="${clarityDS}" var="result">

SELECT 1 from dual

</sql:query>

**Another large benefit of using the <gel:setDataSource/> tag is that it enables connection pooling. This means connections will not be physically created with every query, as is the case with the <sql:setDataSource/> version.**

### <sql.setDataSource/>

As explained above, for Clarity do not use this tag type. However, if needing to connect to some other database (perhaps for an interface) this would then be appropriate.

Note this tag can be quite complex, for instance:

<sql:setDataSource url="jdbc:other\_db:sqlserver://other\_db:1433;DatabaseName=other;serverName=other;InsensitiveResultSetBufferSize=0;ProgramName=Other" driver="com.ca.other.jdbc.sqlserver.SQLServerDriver" user="${OtherUser}" password="${OtherPassword}" var="**non\_clarityDS**"/>

You can have more that one datasource defined, and specify which datasource to use when running sql through the variable “non\_clarityDS” in the above example.

## Example, simple query

 <sql:query var="**result**" dataSource="${clarityDS}">

 select name, unique\_name from srm\_projects

 </sql:query>

The data returned by such a query is made available in what can be considered an array with the name “result” in the above example.

The data contained in the “result” array can then be accessed via the looping structure <core:forEach …> </core:forEach>

<core:forEach trim="true" items="${**result**.rowsByIndex}" var="row">

 <core:forEach var="field" items="${row}">

 <gel:out>${field}</gel:out>

 </core:forEach>

</core:forEach>

## Calling Stored Procedures from GEL

[From KB 7817]
What is the syntax to execute a stored procedure from within GEL Script?

For sql server you do the following, where "v\_prior" is a parameter.

<sql:update dataSource="${clarityDS}">
Z\_AAA\_SP "v\_prior"

</sql:update>

For Oracle, you would do the following, everything within parantheses are examples of parameters that can be passed to your stored procedure:

<sql:update>call XX.Z\_UpdateMessgStatus(${messg\_seqno}, ${cntrl\_seqno}, 'P', 'CLARITY', '${DBUSER}', to\_date('${run\_date}','yyyy-mm-dd hh24:mi:ss'))</sql:update>

# XOG files within GEL

One of the inherent strengths of GEL (and Jelly) is it’s ‘native’ relationship to XML; and of course XOG files are XML So, there are methods within GEL to construct XML to use through XOG.

The tag <gel:parse …..></gel:parse> is designed for xml construction.

The following displays a minimal construct for creating a single user in Clarity
 (User XOG)


In construction this equates to:

Establish xml parsing: <gel:parse ...

 Mandatory XOG information <NikuDataBus ...><Header .../>

 Begin Type of data <Users>

 Give data instance <User ...>

 Data ...

 End data instance </User>

 End Type of data </Users>

 End mandatory XOG information </NikuDataBus>

End xml parsing </gel:parse>

And for multiple instances, it is necessary to repeat or ‘loop’ through the data instance portion of the basic construct.



To achieve this we need to combine many elements:

1. establish gel
2. connect to data source(s)
3. use SQL to get the data
4. setup XOG ‘shell’ ready for data
5. loop through the data returned from SQL
6. format this loop of data into relevant XOG XML format
7. merge the XOG ‘shell’ with the loop of data
8. invoke SOAP
9. execute XOG
10. close down gracefully

These steps can be found in this embedded file: 
eg\_gel\_sql\_xog\_soap.gel.xml

# Soap, XOG & GEL

<gel:parameter var="hostname" default="http://localhost:80//xog"/>

<gel:parameter var="username" default ="admin"/>

<gel:parameter var="password" secure="true"/>

<soap:invoke endpoint="${hostname}">

...

<xog:Username>${username}</xog:Username>

<xog:Password>${password}</xog:Password>

...

# Customization Guidelines

The vendor customization policy forbids writing, inserting or deleting from ‘standard’ Clarity tables. For alteration of ‘standard’ Clarity tables use XOG as this respects the needed application business rules.

Custom attributes are not subject to inbuilt Clarity business rules.

XOG is recommended for all Clarity inserts.

To tables that are outside of Clarity, or not defined and maintained by Clarity SQL may be used for insert/update/delete subject to the restrictions of that database.