ellucian

Banner
Mass Data Update Utility Handbook

Release 8.2
June 2016

Notices

Ellucian
4375 Fair Lakes Court
Fairfax, VA 22033
United States of America

Contents

Introduction

The Mass Data Update Utility is a business tool that institutions can use to quickly and easily develop batch updating processes. That is, the Mass Data Update Utility allows institutions to create their own rules (based on SQL scripts) that can be used to update and insert data for circumstances that are not part of the current Banner functionality.

Warning! The Mass Data Update Utility allows you to easily create processes for modifying data. It is recommended, therefore, that you perform the same thorough testing in a non-production environment that would be needed for any other software development.

Users must understand the principles behind the Mass Data Update Utility, especially when using the auto-populate functionality. It is recommended that you thoroughly test functionality before committing to auto-populating a table.

Typically, there are three types of people who will use the Mass Data Update Utility:

- Rules developer: The rules developer writes the SQL rules for specific tasks.
- Rule set builder: The rule set builder is usually someone with advanced Banner and SQL knowledge. The rule set builder takes the rules that have been developed and combines them to make processes.
- End user: The end user runs the rules and views the results.

You can define rules and parameters that will execute desired actions. For example, you can define rules that will identify candidates for financial aid awards, then calculate candidates' eligibility, and finally create or update the applicant status records.

You can restrict access to process codes, rule codes, and action codes. Restricting access allows data privacy and security to be maintained for users of the Mass Data Update Utility. The security requirements are met by using a combination of Baseline forms and Mass Data Update Utility forms. To restrict access to Mass Data Update Utility data, FGAC VBS business profiles can be associated with Mass Data Update Utility process codes, action codes, and rule codes.

Users can view and manually update any data that was created by the Mass Data Update Utility.

The Mass Data Update Utility interacts with other systems and can act as an interface. This release of the Mass Data Update Utility has the following features:

- Simple the user interface, offering graphical representations of process stages
- Adaptability of rules to other environments with different data structures
- Portability of rules, with standardized XML formats
- Opening to external data sources, such as .CSVs, fixed-length data files, and XML data exports

These features allow you to split a complex process, such as building a report based on several sources of information and iterative stages, into a granular structure with simpler intermediate stages. Furthermore, data structures that might be required to record the result of those steps can be generated on the fly to meet specific reporting needs.

The Mass Data Update Utility also allows partial execution of a process.

You can use any data source that can be queried as a table in a processing task, not only Banner tables but also registered external tables from data files or registered XML documents.

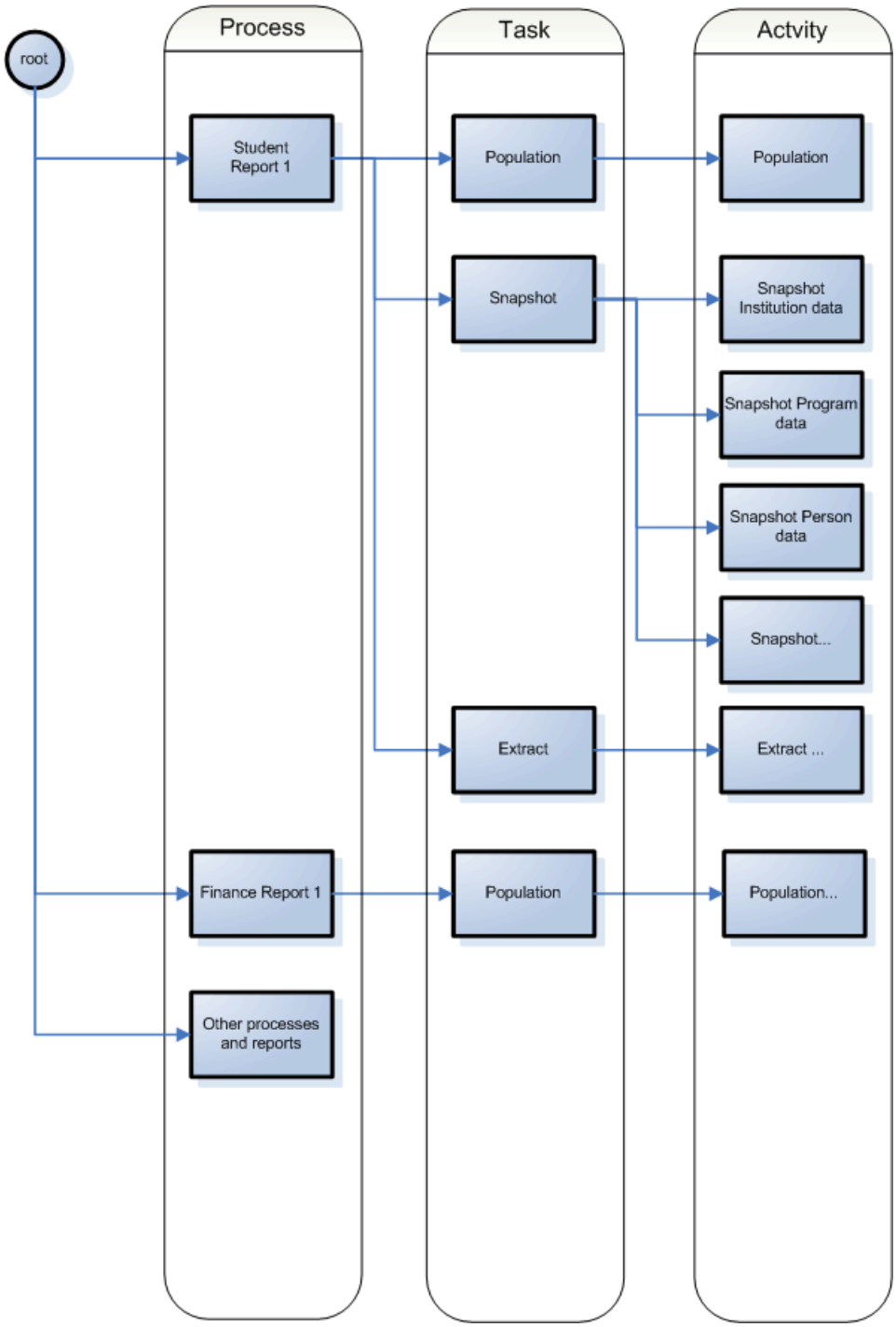
To allow archiving, portability to other databases, or simply communication with other systems, you can also copy, export, and import the setup metadata of process tasks and table structures together with their content.

Note: There are a number of restrictions for this functionality. The processing tasks described below support only the Mass Data Update Utility AUTO_POPULATE and AUTO_DELETE functionality. User-defined actions and other delivered actions are not yet supported.

This document uses the following key terms.

- Process—This represents a set of tasks sharing a common business area. The Mass Data Update Utility security allows you to associate users and user classes to a process. If this functionality is activated, only users assigned to a process will be able to execute it.
- Task—This is a job that is carried out as part of a process, such as creating a population or taking a snapshot of data from the original datasources and gathering them in more synthetic (eventually generated) tables.
- Activity—This is a job that is carried out as part of a task, such as a stage in population, a data copy, or a data transformation. An activity updates a single table and can be reused in several tasks.

The diagram illustrates the structure of processes, tasks, and activities of a reporting activity.



Pre-requisites

The pre-requisites for the initial 8.0 release of the Mass Data Update Utility are listed below.

- General 8.3
- Web Tailor 8.2

Refer to *Banner Release Interdependencies* for the more current dependencies.

Note: Though there is no direct dependency of Web Tailor on MDUU, it uses the same DAD which is used by Web Tailor. Web Tailor 8.2 will be the version supported for MDUU 8.0 in to show the XML and CSV output in the browser.

The users who have PRGN 8.x can upgrade themselves to MDUU. PRGN 7.x will not be supported. The users who have PRGN 7.x should first upgrade themselves to PRGN 8.x and then to MDUU.

Security changes in MDUU

This section describes the security changes for a new MDUU installation or upgrade from PRGN to MDUU.

- New `BAN_MDUU_USER_C` and `BAN_MDUU_ADMIN_C` classes
- Creation of a new import and export directory

Security classes

The population of MDUU users can be divided in two categories: administrative users that can access all MDUU forms and functional users who only run MDUU processes that have been designed for them. Functional users only require access to parts of the MDUU forms.

MDUU introduces two classes that define the type of access a user can have to forms:

- `BAN_MDUU_USER_C` - sets functional user privileges.
- `BAN_MDUU_ADMIN_C` - sets administrative user privileges.

The MDUU objects associated with these classes can be refined through standard Banner Security processes. In addition to forms access restrictions through one of the MDUU security classes, refined security at the MDUU business process level can be defined in the Process Rules Roles (GKAPRRO) form.

In PRGN, PRGN users, irrespective of their role as PRGN administrator or functional user, were granted access to all PRGN forms and processes. This access was granted during a PRGN installation and upgrade processes by associating the security class `BAN_GENERAL_C` to all PRGN users.

During the upgrade from PRGN to MDUU 8.0, MDUU objects are disassociated from the `BAN_GENERAL_C` class, and PRGN users become MDUU users. The MDUU users are then associated to the most restrictive of the new security classes - `BAN_MDUU_USER_C`. By default,

MDUU users will have access to the forms GKAPPLN, GKAPUNV, GKARLOG, GKIPRCT, GKIPRST, GKIPRTR, GKIPSQL, and GKIRSQL. MDUU users will also have access to the process GKPPSQL by default.

Depending on the business processes in place at the institution, the set of objects associated to the `BAN_MDUU_USER_C` class can be modified through the Banner Security Form (GSASECR). For instance, it might be desired to include the GKAPEXE form to the list of objects authorized to a functional user.

Note: By default, no user is associated with the `BAN_MDUU_ADMIN_C` class, which provides access to all MDUU objects. This association needs to be performed by a Banner Security Administrator through the GSASECR form.

Forms/Processes assigned to the security classes

The following table lists the forms and processes assigned to the `BAN_MDUU_USER_C` and `BAN_MDUU_ADMIN_C` classes.

Field	Description
GKAPACT	BAN_MDUU_ADMIN_C
GKAPEXE	BAN_MDUU_ADMIN_C
GKAPEXS	BAN_MDUU_ADMIN_C
GKAPPLN	BAN_MDUU_ADMIN_C
GKAPRRO	BAN_MDUU_ADMIN_C
GKRPRCT	BAN_MDUU_ADMIN_C
GKRPRST	BAN_MDUU_ADMIN_C
GKRRSQL	BAN_MDUU_ADMIN_C
GKRSQPA	BAN_MDUU_ADMIN_C
GKRSQRP	BAN_MDUU_ADMIN_C
GKRSVBA	BAN_MDUU_ADMIN_C
GKAPMLT	BAN_MDUU_USER_C
GKAPUNV	BAN_MDUU_USER_C
GKARLOG	BAN_MDUU_USER_C
GKIPRCT	BAN_MDUU_USER_C
GKIPRST	BAN_MDUU_USER_C
GKIPRTR	BAN_MDUU_USER_C
GKIPSQL	BAN_MDUU_USER_C
GKIRSQL	BAN_MDUU_USER_C

Field	Description
GKPPSQL	BAN_MDUU_USER_C
GKVPACT	BAN_MDUU_USER_C
GKVPRST	BAN_MDUU_USER_C
GKVPSYN	BAN_MDUU_USER_C
GKVPTAS	BAN_MDUU_USER_C
GKVSQPA	BAN_MDUU_USER_C
GKVSQPR	BAN_MDUU_USER_C
GKVSQRU	BAN_MDUU_USER_C
GKVSVAP	BAN_MDUU_USER_C
GKVSVBA	BAN_MDUU_USER_C

Access to a new import and export directory

During the MDUU installation, a directory is created for importing and exporting data.

If a MDUU user wants to use a different directory for importing and exporting data, a new directory must be created. The MDUU user must then be given read and write access to the directory. Afterwards, a database administrator must also create the same directory and then give the MDUU user read and write access to the directory.

Define Processes

This section provides the following step-by-step procedures for defining a Mass Data Update Utility (MDUU) process.

After a process has been defined, you can use it to build a process tree as described in Build Process Trees on page 16 or to build a complex process as described in Build Complex Processes on page 23.

Warning! The Mass Data Update Utility allows you to easily create processes for modifying data. It is recommended, therefore, that you perform the same thorough testing in a non-production environment that would be needed for any other software development.

Users must understand the principles behind the Mass Data Update Utility, especially when using the auto-populate functionality. It is recommended that you thoroughly test functionality before committing to auto-populating a table.

Create process codes

You can create and define a process code. Later, you will use the process codes in other tasks.

About this task

- Attach tasks and activities to the process code when building a process tree
- Attach rules and rule sets to the process code when building a complex process

Procedure

1. Access the Business Rule Process Code Validation Form (GKVSQPR).
2. Enter the code of the process in the **Code** field.
3. Enter a short description of the process in the **Description** field.
4. If the process is required by the system, select the **System Required** check box.
5. Enter the first date the process is to be active in the **Start Date** field.
6. If desired, enter the last date the process is to be active in the **End Date** field.

   Note: The start and optional end dates are for information purposes only. The dates will not restrict when the process code can be used.

7. Save your changes.

Create rule parameter codes

You can define parameters, which are used to build process rules.

Procedure

1. Access the Business Rule Parameter Code Validation Form (GKVSQPA).
2. Enter the code for the parameter in the **Code** field.
3. Enter a short description of the parameter in the **Description** field.
4. Select the data type to be associated with the parameter in the **Data Type** field.
5. Enter the first date the process is to be active in the **Start Date** field.
6. If desired, enter the last date the process is to be active in the **End Date** field.
7. Save your changes.

Define process code parameters

Associate parameters with a process code.

Procedure

1. Access the Business Rules Process Parameters Form (GKRSQPA).
2. Enter the process code with which you want to associate parameters or variables in the **Process Code** field, then go to the next block.
3. Enter the code of the parameter that you want to associate with the process code in the **Parameter Code** field.
4. If the parameter is required by the system, select the **System Required** check box.
5. Save your changes.
6. Repeat steps 3 on page 14 through 5 on page 14 for each parameter you want to associate with the process code.

Set up security rules for process codes

The following procedure explains how to associate FGAC business profiles to process codes. If a business profile and process code are associated, then only the users assigned to the business profile will be able to access the associated process code.

About this task

If no active FGAC business profiles are associated with process codes, then no security restrictions exist for process codes.

Note: It might be easier to copy a row, amend the copy, delete the original row, and then save the changes.

Procedure

1. Access the Process Rules Roles Form (GKAPRRO).
2. **Optional:** In the Key block fields, enter values to filter the records that will be listed in the tabs, then go to the next block.

   Note: If you leave both Key block fields blank, all records will be displayed.
3. Go to the Process Code Profiles tab.
4. Enter the FGAC business profile to be associated with a process code in the **Business Profile** field.
5. Enter the process code to be associated with a business profile in the **Process Code** field.
6. If security restrictions to be in effect for the associated business profile and process code combination, select the **Active** check box.
7. Save your changes.

Build Process Trees

A process “tree” is composed of one or more processing activities arranged in a tree structure. The tree structure is built using the Execution Tree Set-up Form (GKAPEXS) to perform actions related to a business process. All or part of the tree can be executed using the Process Tree Execution Form (GKAPEXE).

Warning! The Mass Data Update Utility allows you to easily create processes for modifying data. It is recommended, therefore, that you perform the same thorough testing in a non-production environment that would be needed for any other software development.

Users must understand the principles behind the Mass Data Update Utility, especially when using the auto-populate functionality. It is recommended that you thoroughly test functionality before committing to auto-populating a table.

The tree is composed of the following elements:

- A single root node, which is the process code of a process defined on GKVSQPR
- One or more branches composed of task codes defined on GKVPTAS

  These branches provide a convenient method for changing the status of all their child-nodes to enable or disable a number of tasks.
- One or more leaf-nodes for each branch

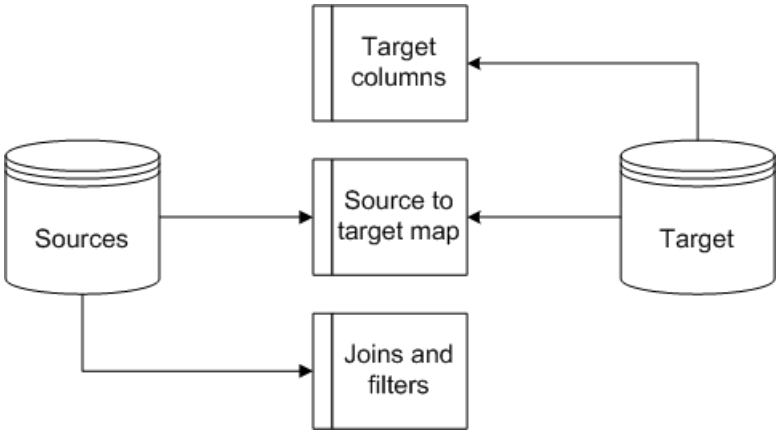
  These are the activities that actually carry out the processing.

An activity is a “leaf-node” of the tree that allows data to be selected from any number of sources and inserted or updated into a single target.

- A processing task selects data from a number of source tables. Any Oracle data table or external table used to represent a .CSV or fixed-length format external file can be used as a source table.
- The target table can be any Oracle table. If the target table does not exist the first time that an activity is created, the Mass Data Update Utility will create a table in a special schema owned by the user PRGNREP. When created, the structure of a table in this schema can be amended by the Mass Data Update Utility by changing or adding column definitions in GKAPACT.
- Any valid subset of the target columns of the target table can be populated.
- Target columns can be mapped from the source column, populated by run-time parameters (bind variables), or derived from SQL expressions or sub-queries using combinations of the above.
- Joins and filters are specified to define the logical relationship between the source tables and the sub-set of records to be processed.

A processing activity is based on the model shown in Figure 1: Structure of a processing activity on page 17.

Figure 1: Structure of a processing activity



The rest of this chapter provides the following step-by-step procedures for process trees:

- Create process and task codes on page 17
- Activity tasks on page 18
- Build model process trees on page 21
- Execute process trees on page 21

Create process and task codes

The following step-by-step procedure explains how to create the codes you need for a process tree.

Procedure

1. If it does not already exist, create and define the process code as described in Create process codes on page 13.
2. Create the task codes you want to use in the process tree as follows.
   a) Access the Process Task Code Validation Form (GKVPTAS).
   b) Enter the code of the task in the **Task Code** field.
   c) Enter a short description of the task in the **Description** field.
   d) Select the process code to be associated with the task in the **Process Code** field.
   e) Save your changes.
   f) Repeat steps 2.b on page 17 through 2.e on page 17 for each task.

Activity tasks

Activities are the processing units (the logic) within process trees, while the trees themselves represent the order in which the system is to apply the processing units.

This section explains how to perform the following tasks.

- Create activity codes on page 18
- Register synonyms on page 18
- Create an activity's logic on page 19

Create activity codes

The following step-by-step procedure explains how to create and define activity codes.

Procedure

1. Access the Process Activity Code Validation Form (GKVPACT).
2. Enter the code of the activity in the **Activity Code** field.
3. Enter a short description of the task in the **Description** field.
4. Select the process code to be associated with the activity in the **Process Code** field.
5. Save your changes.

Register synonyms

The following step-by-step procedure explains how to register synonyms. Only registered synonyms can be used as source of data for activities. This provides a control on the data that can be retrieved using Mass Data Update Utility processes.

Procedure

1. Access the Activity Source Synonym Validation Form (GKVPSYN).
2. Enter the name of the synonym in the **Synonym Name** field.
3. Enter a short description of the table or view represented by the synonym in the **Description** field.
4. Save your changes.

Create an activity's logic

The following step-by-step procedure explains how to create an activity's logic for use in a process tree.

About this task

Warning! The Mass Data Update Utility allows you to easily create processes for modifying data. It is recommended, therefore, that you perform the same thorough testing in a non-production environment that would be needed for any other software development.

Users must understand the principles behind the Mass Data Update Utility, especially when using the auto-populate functionality. It is recommended that you thoroughly test functionality before committing to auto-populating a table.

Procedure

1. Access the Activity Set-up Form (GKAPACT).
2. Enter the activity code in the **Activity Code** field, then go to the next block.

   The system displays the process code associated with the activity.
3. Go to the Source Tables tab to define the locations from where the activity will retrieve data.
4. Enter the name of the registered synonym in the **Synonym of Source/Table View** field.
5. Save your changes
6. Repeat steps 4 on page 19 and 5 on page 19 for each synonym.

   Synonyms can be reused multiple times to allow joins on a single object.
7. Go to the Target Table tab to define the locations in which the activity will store data or from which it will delete data.
8. Enter the table name in the **Target Table Name** field.

   You can enter either an existing table name or a new name. If you enter the name of a table that does not exist, a table will be generated and given that name when the activity is compiled.
9. If you entered a new (non-existing) table name in the **Target Table Name** field, enter a short description of the new table in the **Description** field.
10. Save your changes.
11. Go to the Target Columns tab to define the columns that will be used to store data or that will be used to decide if a record must be deleted by the activity.
12. Enter the column name.

    If you entered an existing table name in the **Target Table Name** field on the Target Table tab, you must enter an existing column name.

    If you entered an existing table name in the **Target Table Name** field and the table has not been generated by the Mass Data Update Utility, you must enter an existing column name. You cannot use the Mass Data Update Utility to extend a baseline table; to extend a baseline table, you must use the Supplemental Data Engine.

If you entered a new (nonexisting) table name in the **Target Table Name** field on the Target Table tab, the column name will be used when the new table is generated by the Mass Data Update Utility.

13. If the column is a new (nonexisting) column, perform the following steps.
    a) Enter a short description of the new column in the **Description** field.
    b) Enter the type of data stored in the column in the **Data Type** field.
    c) If applicable, enter the maximum length of the field in the **Column Length** field.
    d) If applicable, enter the number of characters after the decimal point in the **Scale** field.
    e) If the field can contain a null value, select the **Nullable** check box.
14. Save your changes.
    a) Repeat steps 12 on page 19 through 14 on page 20 for each column in the target table.
15. Go to the Select Parts tab to define how the value to be stored in a target column is to be computed and which columns are used to determine whether a record needs to be updated, created, or deleted.
16. Enter an SQL expression that corresponds to the value to be stored in the column in the **Select SQL** field.
17. If the column is part of the logical primary key of the target table, select the check the **Populate Key** check box.

    When the activity is run, if the values computed for the columns with the Populate Key check box selected match a record, the remaining columns are updated. If no matching record is found, a new record is inserted.
18. If the column is part of the logical primary key of the target table for a purge run, select the **Purge Key** check box.

    When the activity is run with purge, if the values computed for the columns with the **Purge Key** check box selected match a record, the record is deleted from the target table.
19. Save your changes.
20. Repeat steps 16 on page 20 through 19 on page 20 for each target column.
21. Go to the Join/Filter tab to define how the different sources are joined and how selected records are filtered.
22. Enter an SQL expression that restricts the records selected from the sources in the **Filter** field.
23. Enter an SQL expression that defines how the different sources are joined in the **Join** field.
24. Save your changes.
25. Go to the Generated SQL tab.
26. Click  **Compile Activity** to generate the target table and columns if these do not exist in the database and to generate the SQL syntax of the activity.

    The resulting SQL statements can be copied and pasted into any SQL tool for testing, debugging, or fine tuning.

Build model process trees

The following step-by-step procedure explains how to create and define a model process tree. In this procedures, you will create a template processing tree that is inherited by end-users of the process.

Procedure

1. Access the Execution Tree Set-up Form (GKAPEXS).
2. Enter the code of the process for which you want to build a tree in the **Process Code** field, then go to the next block.
3. Perform the following steps to add a node to the tree.
   a) Click an existing node in the left pane to expand it.
   b) Select the template node labeled **SELECT TO INSERT NODE HERE** that corresponds to the location of the new node.
   c) Enter the code of the task or activity for this node in the **Node Code** field.

      The values included on the list of values depend on the depth of the selected node. When a child of the root is selected, the list displayed contains the tasks associated to the root process code (GKVPTAS). When a grandchild of the root is selected, the list displayed contains the activities associated to the root process code (GKVPACT).
   d) Save the newly added node.
   e) Click  **Change Status** to select the appropriate status for the node.
   f) Save your changes.
4. Perform the following steps to delete a node from the tree.
   a) Navigate to the desired node in the left pane and select it.
   b) Highlight the node contained in the pulldown located in the right pane.
   c) Perform Remove Record.
   d) Save your changes.

Execute process trees

The following step-by-step procedure explains how to execute a process tree.

Procedure

1. Access the Process Tree Execution Form (GKAPEXE).
2. Enter the code of the process to be executed in the **Process** field, then go to the next block.
3. Click  **Compile Tree**.

4. Go to the Execution tab.
5. Enter the execution mode in the **Execution Mode** field.

   The `Audit` mode will run the process then rollback the changes, while the `Update` mode will commit the changes to the target tables of the activities.
6. Run the process in one of the following ways.

   - If the process is expected to run for a long time, click **Run From Jobsub**.

     This executes the activity in the secure Job Submission environment and allows you to continue using Banner while the process is running.

   - If the process is expected to run for a short time, click **Run Now**.

     You will not be able to work in your Banner session until the process has completed.
7. When the message is displayed after the process is run, make a note of the run sequence number.

   The run sequence number is used to find the log generated by the process.
8. Verify that the data was changed in the target tables as expected.

   The content of generated target tables can be reviewed in the Universal Viewer Form (GKAPUNV).

Build Complex Processes

This chapter provides the following step-by-step procedures for building complex processes.

Warning! The Mass Data Update Utility allows you to easily create processes for modifying data. It is recommended, therefore, that you perform the same thorough testing in a non-production environment that would be needed for any other software development.

Users must understand the principles behind the Mass Data Update Utility, especially when using the auto-populate functionality. It is recommended that you thoroughly test functionality before committing to auto-populating a table.

Rule set and rule code tasks

This section explains how to perform the following tasks.

Create rule set codes

The following step-by-step procedure explains how to create and define rule set codes, which are used to describe tasks within a process.

Procedure

1. Access the Business Rule Set Code Validation Form (GKVPRST).
2. Enter the code of the rule set in the **Code** field.
3. Enter a short description of the rule set in the **Description** field.

4. Select the mode in which the rule set is to be executed in the **Execute Mode** field. (See Rule set execution modes on page 24 for information about the available modes.)
5. Enter the first date the rule set is to be active in the **Start Date** field.
6. If desired, enter the last date the rule set is to be active in the **End Date** field.
7. Save your changes.

Rule set execution modes

A rule set can be executed in one of the following modes.

- RULE Mode on page 24
- ACTION Mode on page 25

RULE Mode

The RULE mode is suitable for rule sets that are transaction based. When a rule set assigned the RULE mode is executed, it is processed in the following sequence.

Procedure

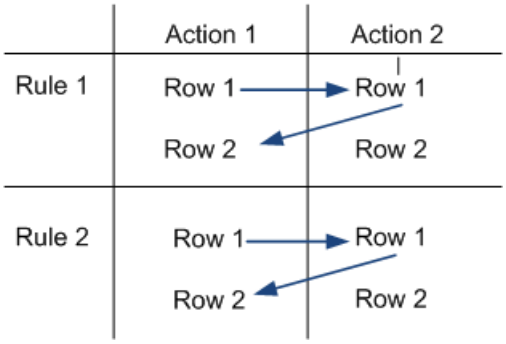
1. Start with the first active, date-effective rule.
2. Get the first row from the rule.
3. Execute all of the active, date-effective actions against the row.
4. When all of the actions are complete, commit the transaction, and get the next row.
5. Repeat the processing with the remaining rows for the rule.
6. Repeat the processing with the remaining rules from the rule set.

Results

Each rule is executed only once. The commit processing in this mode ensures that there are no partially committed transactions for each row returned.

Example

A rule set contains two active rules (Rule 1, Rule 2), which return two rows of data, and two active actions (Action 1, Action 2). When the process is executed in RULE mode, the rules, rows, and actions are processed as shown in the following diagram:



ACTION Mode

When a rule set assigned the ACTION mode is executed, it is processed in the following sequence.

Procedure

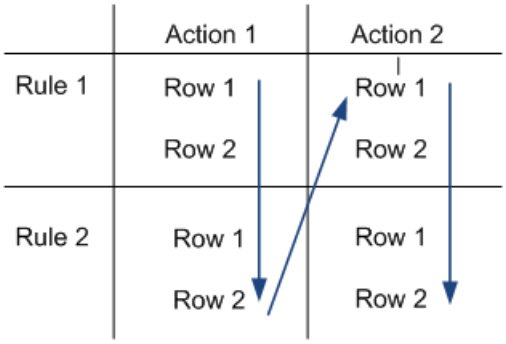
1. Start with the first active, date-effective rule.
2. Get the first row from the rule.
3. Execute the first action against the row.
4. Repeat the processing with the remaining rows from the rule.
5. Commit when the rule is finished.
6. Repeat the rule with the next action.
7. When all of the actions are complete, repeat with the next row.

Results

Each rule is executed multiple times. The results of a previous rule/action combination can be used by the next rule/action.

Example

A rule set contains two active rules (Rule 1, Rule 2), which return two rows of data, and two active actions (Action 1, Action 2). When the process is executed in ACTION mode, the rules, rows, and actions are processed as shown in the following diagram:



Create the rule codes

The following step-by-step procedure explains how to create and define rule codes, which are used to describe common or unique rules that will perform a sub-tasks.

Procedure

1. Access the Business Rule Code Validation Form (GKVSQRU).
2. Enter the code of the rule in the **Code** field.
3. Enter a short description of the rule in the **Description** field.
4. If the rule is required by the system, select the **System Required** check box.
5. Enter the first date the rule is to be active in the **Start Date** field.
6. If desired, enter the last date the rule is to be active in the **End Date** field.
7. Save your changes.

Define the Rule Code Parameters

The following step-by-step procedure explains how to create and define rule code parameters. These parameters will be the default parameters in the Rule Set Rule Parameters block on the Business Process Rule Set Form (GKRPRST) when the process code/rule code combination is the same for both forms.

Procedure

1. Access the Business Rule Parameters Form (GKRSQRP).
2. Enter the process code and rule code combination for which you want to define rule code parameters in the Key block, then go to the next block.
3. Enter the code of the rule parameter in the **Parameter Code** field.

   See Create rule parameter codes on page 14 for instructions for creating rule parameter codes.
4. If the parameter is required by the system, select the **System Required** check box.

5. If desired, enter the default value that is to appear on GKRPRST for this rule in the **Default Value** field.
6. Save your changes.
7. Repeat steps 2 on page 26 through 6 on page 27 for each rule parameter you want associated with the rule code.

Action code tasks

This section explains how to perform the following tasks.

- Create action codes on page 27
- Define the action code parameters on page 28

Create action codes

The following step-by-step procedure explains how to create and define action codes, which are used to specify what action the system is to take for each row of data produced by the SQL rules when a rule set is executed.

Procedure

1. Access the Business Action Code Validation Form (GKVSVBA).
2. Enter the action code in the **Code** field.
3. Enter a short description of the code in the **Description** field.
4. If the action is required by the system, select the **System Required** check box.
5. Enter the name of the package/procedure that is to be associated with the action code in the **Package/Procedure** field.

   Note: Parameters may need to be passed in a user defined package.procedure. The following setup is required to pass dynamic prompted variables to a user defined package.procedure.

   - You must have defined all the parameters on GKVSQPA, GKRSQRP, GKRSQPA, GKRRSQL, GKRSVBA, and GKRPRST.
   - The Package/Procedure call on GKVSVBA must have the bind variables defined in the call itself.

   Note: You can also refer to Article 000005753: 1-17ZGSPS - How can I call an API from MDUU without getting ORA-14551 errors on GKAPEXE, which shows a basic setup for calling a package using a function wrapper and Article 000036221: How to call a custom procedure with parameters from MDUU.
6. Enter the first date the action code is to be active in the **Start Date** field.
7. If desired, enter the last date the action code is to be active in the **End Date** field.
8. Save your changes.

Define the action code parameters

The following step-by-step procedure explains how to create and define the parameter codes and default values for the action codes. These parameters will be the default parameters in the Rule Set Rule Action Parameters block on the Business Process Rule Set Form (GKRPRST) when the process code/action code combination is the same for both forms.

Procedure

1. Access the Business Rule Action Parameters Form (GKRSVBA).
2. Enter the process code and action code combination for which you want to define action code parameters in the Key block, then go to the next block.
3. Enter the code of the rule parameter in the **Parameter Code** field.

   See Create rule parameter codes on page 14 for instructions for creating rule parameter codes.
4. If the parameter is required by the system, select the **System Required** check box.
5. If desired, enter the default value that is to appear on GKRPRST for this action in the **Default Value** field.
6. Save your changes.
7. Repeat steps 2 on page 28 through 6 on page 28 for each action parameter you want associated with the action code.

Auto-populate tasks

This section explains how to perform the following tasks.

- Create auto-populate codes on page 28
- Define auto-populate rules on page 29

Create auto-populate codes

The following step-by-step procedure explains how to create codes used to automatically populate a table when an action is executed.

About this task

Warning! The Mass Data Update Utility allows you to easily create processes for modifying data. It is recommended, therefore, that you perform the same thorough testing in a non-production environment that would be needed for any other software development.

Users must understand the principles behind the Mass Data Update Utility, especially when using the auto-populate functionality. It is recommended that you thoroughly test functionality before committing to auto-populating a table.

Procedure

1. Access the Auto-populate Code Validation Form (GKVSVAP).
2. Enter the name of the database column that is to be automatically populated in the **Code** field.

   The column name thus becomes the auto-populate code.
3. Enter a short description of the auto-populate code in the **Description** (untitled) field.
4. Enter the first date the auto-populate code is to be active in the **Start Date** field.
5. If desired, enter the last date the auto-populate code is to be active in the **End Date** field.
6. Save your changes.

Define auto-populate rules

The following step-by-step procedure explains how to define auto-populate rules.

About this task

Warning! The Mass Data Update Utility allows you to easily create processes for modifying data. It is recommended, therefore, that you perform the same thorough testing in a non-production environment that would be needed for any other software development.

Users must understand the principles behind the Mass Data Update Utility, especially when using the auto-populate functionality. It is recommended that you thoroughly test functionality before committing to auto-populating a table.

Procedure

1. Access the Business Rules Auto-Populate Rules Form (GKRPRCT).
2. Enter the process code, rule code, and action code combination for which you want to define auto-populate rules in the Key block, then go to the next block.

   If you want to view or modify rule for an existing combination, you can click  **Summary** to select the desired record.
3. Enter the number specifying the sequence in which this rule is to be processed when the SQL rule statement is executed in the **Select Position** field.
4. Go to the Auto-Population tab.
5. Enter the table name in the **Table** field.

   The table name must be the same for all rules associated with the record selected in the Key block.
6. Enter the auto-populate code associated with this rule in the **Auto-Populate Code** field.
7. If this column is to be used when the system determines whether to insert a new record or update an existing one when the rule set is executed, select the **Key** check box.
8. Save your changes.
9. Repeat steps 2 on page 29 through 8 on page 29 for each rule.

10. Go to the Display tab.
11. Enter values in the following fields, as needed:
    - **Abbreviation**
    - **Display Sequence**
    - **Order Sequence**
    - **Display Length**
    - **Update Status**
    - **Order By**
    - **Format**
    - **Entity**
    - **Attribute**
12. Save your changes.
13. Repeat steps 11 on page 30 and 12 on page 30 for each rule.

Set up security rules

Use the Process Rules Roles Form (GKAPRRO) to create and maintain the Mass Data Update Utility security rules. From this form, you can associate FGAC VBS business profiles with Mass Data Update Utility process codes, action codes and rule codes.

When process codes, action codes and rule codes are associated with FGAC business profiles, only the users assigned to the FGAC business profile will have access to the associated processes, actions and rule codes. Individual users are assigned to business profiles on the FGAC Business Profile Assignment Form (GOAFBPR). In order to have access to a rule set, a user must have access to its process code and all of the rules and actions the rule set contains.

If no active FGAC business profiles are associated to process codes, action codes or rule codes, then no security restrictions will be in place.

This section explains how to do the following tasks:

- Set up security rules for process codes on page 31
- Set up security rules for action codes on page 31
- Set up security rules for rule codes on page 31
- Set Up Security Rules for Rule Sets on page 32

Set up security rules for process codes

Refer to “Set Up Security Rules for Process Codes”.

Set up security rules for action codes

The Action Code Profiles tab is used to associate FGAC business profiles to process codes and action codes. If a business profile, process code and action code are associated, then only the users assigned to the business profile will be able to access the associated process code/action code combination.

About this task

If no active FGAC business profiles are associated with process codes and action codes, then no security restrictions exist for process code/action code combinations.

A record cannot be changed, except for the **Active** check box, after a record has been saved. To change a record, you must delete it, and add it again with the new data.

Procedure

1. Access the Process Rules Roles Form (GKAPRRO).
2. In the Key block, enter data that will be used to filter the records that will be listed in the tabs.
3. Select **Next Block**.
4. Go to the Action Code Profiles tab.
5. Enter the FGAC business profile to be associated with a process code and action code in the **Business Profile** field.
6. Enter the code of a process to be associated with the specified business profile in the **Process Code** field.
7. Enter the code of an action to be associated with the specified business profile in the **Action Code** field.
8. If security restrictions are to be in place for this business profile, process code, and action code combination, check the **Active** check box.
9. Save your changes.

   Note: After a record has been saved, you cannot change the values in the **Process Code** and **Action Code** fields.

10. Repeat step 5 on page 31 through 9 on page 31 for each business profile you want to associate with the record specified in the Key block.

Set up security rules for rule codes

The Rule Code Profiles tab is used to associate FGAC business profiles to process codes and rule codes. If a business profile, process code and rule code are associated, then only the users

assigned to the business profile will be able to access the associated process code/rule code combination.

About this task

If no active FGAC business profiles are associated with process codes and rule codes, then no security restrictions exist for process code/rule code combinations.

A record cannot be changed, except for the **Active** check box, after a record has been saved. To change a record, you must delete it, and add it again with the new data.

Procedure

1. Access the Process Rules Roles Form (GKAPRRO).
2. In the Key block, enter data that will be used to filter the records that will be listed in the tabs.
3. Select **Next Block**.
4. Go to the Rule Code Profiles tab.
5. Enter the FGAC business profile to be associated with a process code and rule code in the **Business Profile** field.
6. Enter the code of a process to be associated with the specified business profile in the **Process Code** field.
7. Enter the code of a rule to be associated with the specified business profile in the **Rule Code** field.
8. If security restrictions are to be in place for this business profile, process code, and rule code combination, check the **Active** check box.
9. Save your changes.

   Note: After a record has been saved, you cannot change the values in the **Process Code** and **Rule Code** fields.

10. Repeat step 5 on page 32 through 9 on page 32 for each business profile you want to associate with the record specified in the Key block.

Set Up Security Rules for Rule Sets

To have access to a rule set, a user must have access to its process code and all of the rules and actions the rule set contains.

Complete the following tasks for each process code, action code, and rule code contained in the rule set to which you want to assign security rules:

- Set up security rules for process codes on page 14
- Set up security rules for action codes on page 31
- Set up security rules for rule codes on page 31

Create SQL statements for rules

The following step-by-step procedure explains how to create an SQL statement and associate it with a process code/rule code combination. These SQL statements are used to process the sub-tasks (rule codes).

About this task

When the process code/rule code combination is executed from the Process Launch Form (GKAPPLN), the SQL statement is read and executed, and the system return rows of data to be processed by actions (see “Assign Rules and Actions to Rule Sets”).

The only parameters that can be entered on the GKRSSQL form for a process/rule code combination are the parameters attached to the same process code on the GKRSQPA form.

Procedure

1. Access the Business Rules Builder Form (GKRRSQL).
2. Enter the process code and rule code combination for which you want to define rule code parameters in the Key block, then go to the next block.

   If you want to view or modify an SQL statement for an existing combination, you can click  **Summary** to select the desired record.
3. Enter the first date the rule is to be active in the **Start Date** field.
4. If desired, enter the last date the rule is to be active in the **End Date** field.
5. Enter the SQL statement in the **SQL Statement** field.

   You can use the **Table**, **Column**, **Operator** and **Parameter** buttons and fields to help build the statement. Each selection from the buttons and fields is appended to the SQL statement.

   You can enter any parameter/bind variable, however, the SQL will be validated only if attached to the same process code on the GKRSQPA form.

   Note: It is recommended that you use a SQL development tool to create complex SQL statements, which you can then paste into this form.
6. Save your changes.
7. If desired, use the View Parsed Rule option on the Option Menu to view the parsed rule.

Assign rules and actions to rule sets

The following step-by-step procedure explains how to assign sub-tasks (rule codes) and actions to a task (rule set code), and assign the desired parameter values.

About this task

Multiple rules can be added to the process code/rule set combination, which will allow the SQL statements (from GKRRSQL) to be executed in sequence. Actions can also be added to the process code/rule set combination, which will associate a PL/SQL procedure with the rule set.

Rules are used to support SELECT statements (such as “select data using the criteria entered in the SQL statement”). Actions are used to support UPDATE and DELETE statements (such as “update the data in a table”) or used to support other procedures such as calling a Banner API. The design of the software allows clients to develop local actions to support their own processing needs.

The parameters defined on the Business Rule Parameters Form (GKRSQRP) and Business Rule Action Parameters Form (GKRSVBA) will be the default parameters on GKRPRST. The default values can be overridden.

When the desired rules and actions are added to a process code/rule set combination, the GKAPPLN form can be used to execute the rule set and the attached rules and actions, or the rule can be executed by the APIs (see APIs on page 44).

Procedure

1. Access the Business Process Rule Set Form (GKRPRST).
2. Enter a value in the **Process Code** field and the **Rule Set** field, and go to the next block.
3. Assign rules to the process/rule set combination as follows.
   a) In the Rule Set Rules block, enter a value in the **Run Sequence** and **Rule** fields.
   b) If needed, select the rule for which you want to adjust the parameter default values, and go to the next block.

      A rule will have parameters assigned to it if parameters were assigned to the same process/rule code combination on the GKRSQRP form.
   c) If needed, in the Rule Set Rule Parameters block, change the value in the **Default Value** field.
   d) Save the record.
4. Assign actions to the process/rule set combination.
   a) In the Rule Set Actions block, enter a value in the **Seq** and **Action** fields.
   b) If needed, select the action for which you want to adjust the parameter default values, and select Next Block.

      An action will have parameters assigned to it if parameters were assigned to the same process/action code combination on the GKRSVBA form.
   c) If needed, in the Rule Set Action Parameters block, change the value in the **Default Value** field.

d) Save the record.

Define the column display

Use the Process Rule Column Definition Form (GKRPRCT) to define the columns that will be used during processing. The GKRPRCT form is also used to determine how the GKRPWRK data will be displayed in the Working Results window on the Process Transaction Maintenance Form (GKAPMLT).

Two types of column display can be defined:

- Define the columns used by actions on page 35
- Define the columns used by GKAPMLT on page 36

Define the columns used by actions

The following step-by-step procedure explains how to define the columns to be used by the actions, for example the auto-populate columns. The task allows auto-populate and auto-delete to construct the appropriate insert, update, and delete statements.

About this task

The PL/SQL process action (`GKKPSQL.AUTOPOPULATE`) will insert a new record or update an existing record. See AUTO_POPULATE on page 40 for details.

The fields on the Auto-Population tab provide a mapping, between the columns returned by the select of a SQL rule, and the columns designated as auto-populate using the delivered AUTO_POPULATION process action. In addition, these rules are used to provide a logical association between rule data and action data for other delivered actions and for client-defined actions.

Note: The AUTO_POPULATE action (`GKKPSQL.AUTOPOPULATE`) enforces strict type-validation, it does not carry out implicit type conversion. If the data type of the column referenced by the **Select Position** field (`GKRPRCT_SELECT_POSITION`) does not agree with the data type in the **Auto-Populate Code** field (`GKRPRCT_SVAP_CODE`), then a run-time exception error such as `ORA-01722: Invalid Number` will be raised.

Procedure

1. Access the Business Rules Auto-Populate Rules Form (GKRPRCT).
2. Enter the process code, rule code, and action code combination for which you want to define auto-populate rules in the Key block, then go to the next block.

   If you want to view or modify rule for an existing combination, you can click  **Summary** to select the desired record.
3. Enter the number specifying the sequence in which this rule is to be processed when the SQL rule statement is executed in the **Select Position** field.

4. Go to the Auto-Population tab.
5. Enter the table name in the **Table** field.

   The table name must be the same for all rules associated with the record selected in the Key block.
6. Enter the auto-populate code associated with this rule in the **Auto-Populate Code** field.
7. If this column is to be used when the system determines whether to insert a new record or update an existing one when the rule set is executed, select the **Key** check box.
8. Save your changes.
9. Repeat 2 on page 35 steps through 8 on page 36 for each rule.

Define the columns used by GKAPMLT

This task is used to define the display columns for the GKAPMLT form. Only data written to the GKRPWRK table can be made visible on the GKAPMLT form.

About this task

The information entered on the Display tab determines how the data will be displayed on the Process Transaction block on the GKAPMLT form when the GKPPSQL process is run (which executes the rule set).

For data to be displayed on the Process Transaction block on the GKAPMLT form, values must be entered in the **Abbreviation**, **Display Sequence** and **Order Sequence** fields.

Note: The rule set execution environment uses the GKRPRCT form along with the process, rule and action to find out how to handle the current row.

The GKAPMLT form uses the process code, rule code and action code to find the column definitions to know what to display and how. The GKAPMLT form also uses the process code, rule code and action code as a filter when retrieving rows from the GKRPWRK table.

Procedure

1. Access the Business Rules Auto-Populate Rules Form (GKRPRCT).
2. Enter the process code, rule code, and action code combination for which you want to define auto-populate rules in the Key block, then go to the next block.

   If you want to view or modify rule for an existing combination, you can click  **Summary** to select the desired record.
3. Go to the Auto-Population tab and define the values as described in Define the columns used by actions on page 35.
4. Go to the Display tab.
5. Enter values in the following fields.
   - **Abbreviation**
   - **Display Sequence**

- **Order Sequence**
- **Display Length**
- **Update Status**
- **ASC/DESC**
- **Format**
- **Entity**
- **Attribute**

  Note: Values must be entered in the **Abbreviation**, **Display Sequence**, and **Display Length** for the item to be visible.

6. Save your changes.

Execute a rule set

The following step-by-step procedure explains how to specify the parameter values and execute the rule set. Executing a rule set runs the SQL rules (where the SQL is associated with the rule code on the Business Rules Builder Form [GKRRSQL]), and carries out the action(s) for each row of data returned.

About this task

The rules and actions are assigned to a process/rule set combination from the Business Process Rule Set Form (GKRPRST).

The Business Rule Process (GKPPSQL) is used to execute the process/rule set combination. GKPPSQL can be run from this form, from the Process Submission Controls Form (GJAPCTL), or by using the `gkkpsql.API_ExecuteRuleset` API (see gkkpsql.API_ExecuteRuleset on page 46).

Note: Dynamic parameters are built into a ruleset when GKPPSQL launches GJAPCTL using the Run From Jobsub option on GKAPEXE or GKAPPLN. An error can occur when attempting to execute GKPPSQL directly from GJAPCTL without first building the ruleset on GKAPEXE or GKAPPLN.

GKPPSQL is intended to launch GJAPCTL when users execute using the Run From Jobsub option available on GKAPEXE or GKAPPLN as dynamic parameters are built into a ruleset for the user at that point. ORA-01008: not all variables bound errors can occur if users attempt to execute GKPPSQL directly from GJAPCTL without first building the ruleset on the MDUU forms GKAPEXE or GKAPPLN.

Procedure

1. Access the Process Launch Form (GKAPPLN).
2. Enter values in the **Process Code**, **Rule Set** and **Effective Date** fields, and go to the next block

If you want to view or modify rule for an existing combination, you can click **Summary** to select the desired record.

3. From the Process Launch Parameter tab, enter values in the following fields:
   - **Execution Mode**
   - **Exception Mode**
   - **Diagnostic Severity**

     See Process Launch Parameters tab on page 81 for an explanation of what parameters are displayed.
4. If necessary, enter or change any parameter values.
5. Click **Run From Jobsub** or **Run Now**.

Results

When the rule set is completed successfully, a pop-up window displays the process sequence number (under which the diagnostic data is stored in GKARLOG) and confirms the process and rule set codes.

If an unhandled exception is encountered causing the rule set to fail, then an error message is displayed showing the sequence number.

View the output for a process/action/rule combination

The following step-by-step procedure explains how to view, add, or update data after the Business Rule Process (GKPPSQL) has been run.

About this task

The columns that are displayed are those specified in the Business Rules Auto-Populate Rules Form (GKRPRCT).

Procedure

1. Access the Process Transaction Maintenance Form (GKAPMLT).
2. Enter values in the **Process Code**, **Rule**, and **Action** fields, and select Next Block.

   The Filter block is populated with the data from the GKRPRCT table for the selected process/rule/action combination.
3. Enter values in the following fields to specify the information that will be available from the Process Transactions block:
   - **Query Value**
   - **Include Null**
   - **Display**
   - **Order Priority**

- **ASC/DESC**

4. Go to the next block to view the data that resulted from when the rule set was executed.

   The columns that are displayed are based on what was entered for the process/rule/action combination on the GKRPRCT form, and based on the data supplied in the Filter block.

5. Change the data if necessary.
6. Click **Extract Data** to extract the data to your desktop using a modified version of the Help Data Extract functionality.

   Note: If you changed any data, you might want to bring that changed data back into Banner. You will need to create another rule set to bring the data back into Banner.

Action Codes

This chapter describes the following action codes, which will launch a task (rule set) and sub-tasks (rules) within a process.

AUTO_POPULATE

The PL/SQL process action (`GKKPSQL.AUTOPOPULATE`) will insert a new record or update an existing record.

The action will execute '`if exists then update else insert`' logic when processing a rule set with auto-population. The AUTO_POPULATE process action will use the column rules for the current process / rule /action to either insert a new record or update an existing record using the column rules in GKRPRCT with the merge indicator to define the key columns as described below.

It is assumed that each row returned when executing a rule against an action will result in an insert / update to a single Banner table.

The AUTO_POPULATE process action will execute the following logic for each row returned by the rule / action combination:

- Construct a where-clause based on the GKRPRCT rules for the current process rule / action as follows:

  ```
  Set the value of the where-clause to 'WHERE 'For each column definition
  rule for the current rule / action where GKRPRCT_MERGE_IND <> 'N',
  append the value GKRPRCT_SVAP_CODE, the text ' = '; the value returned
  in the rule cursor; and the text ' AND '.
  ```

- Execute the following logic:

  ```
  If a where-clause has been constructed, thenselect row from
  GKRPRCT_ENTITY for update where …if row found, then update table set
  columnlist = VALUE where-clause else insert into table
  ```

Note: The AUTO_POPULATE action makes no assumptions and has no meta-knowledge about the tables being updated other than that provided in the GKRPRCT form. The person building the rule-set is responsible for ensuring that all Oracle constraints, Banner constrains and local business constraints are satisfied.

In particular, the AUTO_POPULATE action does not automatically populate columns in the GKRPWRK table when inserting rows into that table.

AUTO_DELETE

The process action will allow the user to delete Oracle data (and insert and modify data).

The `GKKPSQL.AUTODELETE` process action will use the key rules in the Process Rule Column Definition Form (GKRPRCT) to delete each row as it is processed. A parameter will allow for changes to be either committed or rolled-back. The process will optionally write a log row for each delete statement that was processed. Rules will now be able to be executed in debug mode.

The enhanced column definition rules will be used to dynamically construct a `DELETE FROM` statement constructing the where clause in the same way that the statement is constructed for the AUTO_POPULATE process action.

SQL Functions

A SQL function is part of the SQL environment that allows you to calculate a returned value as part of a SQL Select statement.

For example, with `select lower('ABC'), ...`, the `lower('ABC')` is a function that converts characters to lower case. Oracle provides many built-in functions (such as `lower`).

Oracle also allows user-defined functions. Some user-defined functions are provided, which means that although the functions are intended for the Mass Data Update Utility, the functions could be used in an SQL SELECT statement for other reporting purposes.

The following SQL functions have been provided:

- COLLATETEXT on page 42
- MERGEVARIABLES on page 42

COLLATETEXT

The `GKKPSQL.COLLATETEXT` function can be used by any SQL rule within a reporting environment or the Mass Data Update Utility. The main aim is to return rows of text (or other items) into a single `CLOB` or `LONG` data item.

An example would be to return a number of rows of program text from SMRPCMT into a single text paragraph.

This function returns a `CLOB` data item.

The function parameters are as follows:

- `SQL_STATEMENT`: A `CLOB` data item that contains an SQL statement.
- `DELIMITER`: A `VARCHAR` data item that contains a delimiter, which can be null.

The `GKKPSQL.COLLATETEXT` function will use dynamic SQL to bind and execute the `SQL_STATEMENT` parameter, and append each row returned into the data item to be returned. The function will append the `DELIMITER` parameter value to each row to separate the text. The `DELIMITER` value is not applied to the end of the final row that is returned.

MERGEVARIABLES

The `GKKPSQL.MERGEVARIABLES` function returns the results, which occurred when a set of variable data was merged with the boilerplate text.

This function returns a `CLOB` data item that is the result of merging the variable string into the boilerplate text.

The function parameters are as follows:

- VARIABLE_STRING: A list of the variables extracted from the database. Each of the variables is separated by the delimiter specified in the VARIABLE_DELIMITER parameter. An example of a variable string is @0000205||Rick Burne||Academic Year 2005/6||01 September 2006.
- VARIABLE_DELIMITER: A string containing the variable that is used to separate the items in the variable string. An example of a variable delimiter is ||.
- BOILERPLATE: The string that contains the boilerplate text. An example of boilerplate text is "Dear <<2>> <p> This is to confirm that you will be starting your studies in <<3>>."
- INSERT_PATTERN: The pattern that is used to determine the insertion points within the boilerplate text. An example of a pattern is <<#>>.

The data is parsed in the following way:

- The variable delimiter is used to extract each variable data element in turn.
- The insertion points are replaced in the boilerplate for the current variable with the value of the variable.
- Stops when no variables are left.

APIs

The following APIs assist in developing local actions and rules.

`gkkpsql.API_ExecuteRule`	Evaluates a stored rule. For more information, see gkkpsql.API_ExecuteRule on page 45.
`gkkpsql.API_ExecuteRuleset`	Allows a rule set to be executed externally. For more information, see gkkpsql.API_ExecuteRuleset on page 46.
`gkkpsql.API_RetrieveActionCode`	Returns the current action being processed. See Access miscellaneous information on page 50.
`gkkpsql.API_RetrieveExecutionMode`	Determines whether the rule is executed in UPDATE or AUDIT mode. See Access the execution mode of the ruleset on page 48.
`gkkpsql.API_RetrieveRulesetParameter`	Returns the value of a user entered parameter.
`gkkpsql.API_RetrieveColumnValue`	Returns the value of a column from the current row according to its column name or abbreviation. See Access the last row of data retrieved on page 49.
`gkkpsql.API_RetrieveRunSequence`	Returns the value of the current run sequence number. See Access miscellaneous information on page 50.
`gkkpsql.API_RetrieveProcessCode`	Returns the value of the current process code (`GVSQPR_CODE`). See Access miscellaneous information on page 50.
`gkkpsql.API_RetrieveRuleCode`	Returns the current rule being processed. See Access miscellaneous information on page 50.
`gkkpsql.API_RetrieveRulesetCode`	Returns the value of the current rule set code (`GKVPRST_CODE`). See Access the ruleset parameters on page 50.
`gkkpsql.WriteDiagnosticToDB`	Writes diagnostic message to GKRRLOG. See Write diagnostic information to the GKARLOG table on page 50.

Examples of how to use these APIs can be found in `gkkpsql.UserActionTemplate`, which is provided as a guide for developing local actions. See the *API Developer Guide* for more information on how to use APIs. Also see Sample PL/SQL procedure on page 51 for a sample PL/SQL procedure that can be used as a template.

gkkpsql.API_ExecuteRule

This API is used to evaluate a stored rule.

The API will do the following:

- Find the fist valid, active, date effective sequence in the GKRRSQL rule.
- Bind the variables held as parameter-value pairs in to the rule.
- Execute the SQL statement to find the first non-null value in the first column

  If a non-null value is not found, then execute the next valid, active, date effective rule sequence.

  Continue until a non-null value is found or there are no rules left.

  If no value is found, then return null.

  Otherwise, return the value of the first column in the first row found.

The parameters are as follows.

Parameter	Description	Comments
1	Process code that identifies the process associated with the rule.	GKRRSQL_SQPR_CODE
2	Rule code that identifies the SQL rule.	GKRRSQL_SQRU_CODE
3	Parameter string	Parameter name/value pairs that will be bound to the variables in the rule.*
4	Delimiter	Character(s) to delimit the rule set parameter strings.*

*The parameter string is made up of a number of parameter/value pairs separated by the delimiter. For example, if the delimiter is '| |' and the parameter string is:

```
Gkkpsql.API_ExecuteRule ('HESA', 'COMDATE', 'PIDM||1234||ADMIT_TERM||
200304||MAJOR||'. '||')
```

then the following values will be available for binding the rules within a rule set:

Bind Variable	Value
||PIDM	1234
||ADMIT_TERM	200304
||MAJOR	Null

An example of evaluating a rule in this way is provided in `gkkpsql.ExecuteRuleTemplate`.

gkkpsql.API_ExecuteRuleset

This API is used to allow a rule set to be executed externally.

The API will do the following:

- Get the next sequence number
- Insert the parameters into GKRPSQL
- Execute the process rule set

The following data will be passed to the Mass Data Update Utility to launch a rule set:

- Process code
- Rule set code
- Effective date
- Execution mode
- Exception mode
- Diagnostic severity
- Parameters

The parameters are as follows:

Parameter	Description	Comments
1	Mass Data Update Utility process code to identify the rule set.	`GKRPRST_SQPR_CODE`
2	Rule set code to identify the rule set.	`GKRPRST_PRST_CODE`
3	Execution mode	AUDIT or UPDATE
4	Exception mode	Log and Abort or Log and Continue
5	Severity	Positive integer
6	Rule set parameter string	Values to bind into the rules.*
7	Delimiter	Character(s) to delimit the rule set parameter strings*.

*The rule set parameter string is made up of a number of parameter/value pairs separated by the delimiter (parameter 7). For example, if the delimiter is ':' and the parameter string is:

```
PIDM:123:START_DATE::TERM_CODE:200506
```

then the following values will be available for binding the rules within a rule set:

Bind Variable	Value
:PIDM	123
:START_DATE	Null
:TERM_CODE	200506

Rule sets may be protected by the security arrangements as set up on the Process Rules Roles Form (GKAPRRO).The API (API_ExecuteRuleset) will assume that these security restrictions are managed externally. The API does not enforce these security restrictions.

Because a rule set may take some time to execute, if a user warning is required, it must be provided by the calling process. A warning will not be provided by the API (API_ExecuteRuleset).

The Mass Data Update Utility will raise an Oracle exception if the API does not execute successfully. The end-user layer must be capable of handling these exceptions in the following ways:

- Rollback all database changes
- Force the end-user to either change the data or rollback
- Ignore the exception

The API (API_ExecuteRuleset) will create a unique sequence number, and will report this number along with any processing exception.

The rule set will always be executed by the user session, not through Job Submission.

The following example illustrates how SQLPLUS can be used to allow the API to be called from the server operating system command line. A similar example procedure is included in `gkkpsql.ExecuteRuleSetTemplate`.

```
1. Create a script called ruleset.sql
set serveroutput on
set verify of
DECLARE
ProcessCode        VARCHAR2(30);
RulesetCode        VARCHAR2(30);
ExecutionMode      VARCHAR2(1);
ExceptionMode      INTEGER;
DiagnosticSeverity NUMBER;
RulesetParameters  VARCHAR2(1000);
Delimiter          VARCHAR2(2);
RunSequenceNum     INTEGER;
Message            VARCHAR2(2000);
TimeStamp VARCHAR2(50) := G$_DATE.GET_NLS_DATE_FORMAT || ' HH24:MI:SS';
BEGIN
  dbms_output.enable(1000000);
-- Retrieve command line parameters
ProcessCode := '&1';
RulesetCode := '&2';
ExecutionMode := '&3';
ExceptionMode := &4;
DiagnosticSeverity := &5;
RulesetParameters := '&6';
```

```
Delimiter := '&7';
```

```
dbms_output.put_line( G$_NLS.Get( 'X', 'SQL',
  'invoked at %01%',
  to_char(sysdate,TimeStamp) ) )
-- Invoke the Mass Data Update Utility
gkkpsql.API_ExecuteRuleset( ProcessCode,
                            RulesetCode,
                            RulesetParameters,
                            Delimiter,
                            ExecutionMode,
                            ExceptionMode,
                          DiagnosticSeverity,
                            RunSequenceNum,
                            Message );
IF Message IS NULL
THEN
  dbms_output.put_line( G$_NLS.Get('X', 'SQL',
      'Successful completion at %01%. Run Sequence = %02%',
    to_char(sysdate,Timestamp), RunSequenceNum ) );
ELSE
  dbms_output.put_line( G$_NLS.Get('X', 'SQL',
    'Unsuccessful completion at %01%. Run Sequence = %02%',
    to_char(sysdate,Timestamp), RunSequenceNum ) );
  dbms_output.put_line( substr(Message,1,255) );
END IF;
END;
/
quit------- ruleset.sql finishes here ---------
2. Call from the unix prompt (cron job)
sqlplus user/password@connect_string @ruleset ... parameters ...
```

Using APIs

Users can use APIs to perform the following.

- Access the execution mode of the ruleset on page 48
- Access the last row of data retrieved on page 49
- Access the ruleset parameters on page 50
- Access miscellaneous information on page 50
- Write diagnostic information to the GKARLOG table on page 50

Access the execution mode of the ruleset

The execution mode of the ruleset is retrieved by calling the following API routine.

```
PROCEDURE gkkpsql.API_RetrieveExecutionMode( pExecutionMode OUT VARCHAR2 );
```

Either AUDIT or UPDATE will be returned.

Access the last row of data retrieved

The last row of data retrieved is stored in a global structure: `GKKPSQL.gRowDetails`. `gRowDetails` is a PL/SQL table of column data, and can only be accessed from the following supplied API routine.

```
PROCEDURE gkkpsql.API_RetrieveColumnValue( pColumnNumber  IN  PLS_INTEGER DEFAULT NULL,
                                           pColumnName    IN  VARCHAR2    DEFAULT NULL,
                                           pColumnAbbr    IN  VARCHAR2    DEFAULT NULL,
                                           pAPIColumn     OUT API_ColumnType,
                                            pErrorMessage  OUT VARCHAR2 )
```

The definition of `API_ColumnType` is as follows:

```
TYPE API_ColumnType is RECORD ( Varchar2Value VARCHAR2(4000),
                                NumberValue   NUMBER,
                                LongValue     LONG,
                                RowidValue    ROWID,
                                DateValue     DATE,
                                ClobValue     CLOB,
                                 BlobValue     BLOB );
```

The column values can be retrieved by specifying one of three things:

- Column Number: The column's position in the `SELECT` clause.
- Column Name: For example `gkrpwrk_v01`.
- Column Abbreviation: The column's abbreviation as specified in the appropriate GKRPRCT entry for the current rule/action being processed.

The data type of the column determines which of the output variables is populated. For example, a date will be stored in `pAPIColumn.DateValue`.

Example

The following is an example of a note parameter name convention:

```
API_RetrieveColumnValue( pColumnNumber => 3,
                         pAPIColumn => MyColumn,
                         pErrorMessage => MyMessage );
MyDate := MyColumn.DateValue;
API_RetrieveColumnValue( pColumnName => 'NameOfMyColumn',
                         pAPIColumn => MyColumn,
                         pErrorMessage => MyMessage );
MyNumber := MyColumn.NumberValue;
```

If `pErrorMessage` has a null value, then the column value has been retrieved successfully.

Access the ruleset parameters

The ruleset parameters are retrieved by calling the following API routine.

```
PROCEDURE gkkpsql API_RetrieveRulesetParameter( pParameterName  IN  VARCHAR2,
                                                pParameterValue OUT VARCHAR2,
                                                 pErrorMessage   OUT VARCHAR2 )
```

If `pErrorMessage` has a null value, then the parameter value has been retrieved successfully.

Access miscellaneous information

Access a run sequence number by calling the following API: `gkkpsql.API_RetrieveRunSequence`.

Access a process code by calling the following API: `gkkpsql.API_RetrieveProcessCode`.

Access a ruleset code by calling the following API: `gkkpsql.API_RetrieveRulesetCode`.

Write diagnostic information to the GKARLOG table

Diagnostic information can be written to the GKARLOG table using the following API.

```
PROCEDURE gkkpsql.WriteDiagnosticToDB( pDiagnosticSeverity IN INTEGER,
                                       pDiagnosticInfo     IN VARCHAR2);
```

The possible diagnostic severities are as follows:

- 10: Detailed diagnostics
- 20: Performance diagnostics
- 30: Progress messages
- 40: Exception warnings
- 50: Fatal exceptions

All diagnostics greater than the value supplied to `WriteDiagnosticToDB` are written to GKARLOG.

Note: `WriteDiagnosticToDB` acts as an autonomous transaction and does not affect the current transaction.

Sample PL/SQL procedure

The sample PL/SQL procedure can be used as a template by clients who want to develop their own business actions for use with the Mass Data Update Utility. A shorter version of this sample procedure is delivered as seed data in the PL/SQL package GKKPSQL.

Note: If you develop a local action that calls Banner APIs or other code that includes an Oracle Commit then the AUDIT / UPDATE execution mode will not be honored.

```
PROCEDURE UserActionTemplate
IS
  MyExecutionMode  VARCHAR2(6);
  MyColumn         gkkpsql.API_ColumnType;
  MyNumber         NUMBER;
  MyMessage        VARCHAR2(4000);
  MyParameterValue VARCHAR2(4000);
  MyRunSequence    NUMBER;
  MyProcess        gkvsqpr.gkvsqpr_code%TYPE;
  MyRuleset        gkvprst.gkvprst_code%TYPE;
  API_Exception    EXCEPTION;
BEGIN
  -- Examples
/* Determine ruleset execution mode: AUDIT or UPDATE
   This allows the action, for example to decide whether or not to commit transactions
   The procedure call below assigns the execution mode to the variable MyExecutionMode */
   gkkpsql.API_RetrieveExecutionMode( MyExecutionMode );
/* Values from the current row can be retrieved by column name, select postion or alias
   Retrieve a column with a NUMBER data type by supplying the column name
   The procedure calls below assign the value of the value within the row to the variable MyColumn
   using each of these methods
   If the procedure succeeds then the value in the MyMessage variable will be null
   a) retrieve the value by Column name 'GKRPWRK_N01' - as defined in GKRPRCT_SVAP_CODE*/
   gkkpsql.API_RetrieveColumnValue( pColumnName   => 'GKRPWRK_N01',
                                    pAPIColumn    =>  MyColumn,
                                    pErrorMessage => MyMessage );
/* b) retrieve the value in column 7 - as defined in GKRPRCT_SELECT_POSITION */
   gkkpsql.API_RetrieveColumnValue( pColumnNumber   => 7,
                                    pAPIColumn    =>  MyColumn,
                                    pErrorMessage => MyMessage );
/* c) retrieve the value in column with abbreviation 'RECOMMEND' - as defined in GKRPRCT_SVAP_abbr */
   gkkpsql.API_RetrieveColumnValue( pColumnNumber   => 'RECOMMEND',
                                    pAPIColumn    =>  MyColumn,
                                    pErrorMessage => MyMessage );
/* check that the previous call succeeded
```

```
  IF MyMessage IS NULL THEN  -- success
    MyNumber := MyColumn.NumberValue;
  ELSE
    RAISE API_Exception;
  END IF;
/* Assign the value of a runtime parameter called 'MyParameterName' to variable MyParameterValue */
   gkkpsql.API_RetrieveRulesetParameter( 'MyParameterName',
                                         MyParameterValue,
                                         MyMessage );
   IF MyMessage IS NOT NULL THEN  -- failure
     RAISE API_Exception;
   END IF;
/* The following shows how to retrieve the value of a number of other run-time environment values which
   can be used within a locally defined action
   Run Sequence Number */
   MyRunSequence := gkkpsql.API_RetrieveRunSequence;
/* Process Code */
   MyProcess := gkkpsql.API_RetrieveProcessCode;
/* Rule Set Code */
```

```
   MyRuleset := gkkpsql.API_RetrieveRulesetCode;
/* The procedure below will write a an entry to the diagnostic table (GKRRLOG)
   with a security of 30 */
   gkkpsql.WriteDiagnosticToDB( 30,  -- Progress message
                               'Progress message written from UserActionTemplate' );
EXCEPTION
  WHEN API_Exception THEN
    -- take appropriate action
    NULL;
END;
```

Process

The following process is part of the Mass Data Update Utility release.

Business Rule Process (GKPPSQL)

This process is used to execute a process/rule set/action function combination. The combination is used to group students and store values that will identify the students in a specific group, which can be used, for example, to make progression recommendations or to update student records.

The process can be run from the Process Submission Control Form (GJAPCTL) or the Process Launch Form (GKAPPLN).

Note: Dynamic parameters are built into a ruleset when GKPPSQL launches GJAPCTL using the Run From Jobsub option on GKAPEXE or GKAPPLN. An error can occur when attempting to execute GKPPSQL directly from GJAPCTL without first building the ruleset on GKAPEXE or GKAPPLN.

The rule launching sequence is run based on the execution mode (RULE or ACTION) selected on the Business Rule Set Code Validation Form (GKVPRST).

RULE Mode

This mode is used when the value is `RULE` in the **Execute Mode** field on the GKVPRST form.

For each rule (in GKRPRST) that the user has permission to launch, the process will retrieve a row from each rule sequence (in GKRRSQL). The process will then execute each action (in GKRPACT) for which the user has permission.

Executing the RULE mode means the following:

- Each SQL rule is executed only once
- All of the actions for the rule can be considered as a single transaction and committed together
- The results of a previous rule cannot be used within the rule set

ACTION Mode

This mode is used when the value is `ACTION` in the **Execute Mode** field on the GKVPRST form.

The process retrieves each action (GKRPACT) that the user has permission to launch, then each rule (in GKRPRST), and then the row (s) from rule sequence (in GKRRSQL), Then the process executes the action against each row that was returned for the rule.

Executing the ACTION mode means the following:

- The SQL rules are executed multiple times

- The results of one entire rule/action combination can be used by the next rule/action combination

When the process is run, the following occurs:

- A one-up sequence number is generated to identify the parameter set.

The following diagnostic information is written to the GKRRLOG table:

Event	Log	Severity
Start rule set	Start rule set: sequence SEQUENCE; Process code: process rule code, date/time stamp	20
Start processing a new rule	Start rule: rule code, date/time stamp	20
Start Processing a new rule sequence	Start rule sequence: rule code / sequence, date/time stamp	20
Complete processing a rule sequence	Completed rule sequence: rule code / sequence; nnnn rows processed, date/time stamp	20
Complete processing a rule	Completed rule: rule code, date/time stamp	20
Execute each action	Executing action: action code, with parameters calling sequence	10
Completed rule set	Completed rule set: sequence: SEQUENCE; Process code: process rule code, date/time stamp	20
Unhandled exception	Exception raised in ‘procedure / function’: procedure name exception 'as much context as needed	30
Handled exception	Oracle error message and context	40

The parameters below are used when the rule set is executed through Banner Job Submission using the GJAPCTL form. Rule set execution through Banner Job Submission is only supported if the job is initiated from the GJAPCTL form through the GKAPPLN form.

Parameters	Name	Required?	Description	Values
	Process Code	Yes	SQL process code with which the set of SQL	Business Rule Process Code

Parameters	Name	Required?	Description	Values
			statements will be associated.	Validation Form (GKVSQPR)
	Ruleset Code	Yes	Process rule set code for dynamic SQL processing.	Business Rule Set Code Validation Form (GKVPRST)
	Run Sequence Number	Yes	Indicates the sequence number that will be applied to the results of the process.	
	Execution Mode	Yes	Execution mode is used by the delivered actions. Locally developed actions can make use of this to meet specific institutional requirements.	AUDIT (default) The rule set is executed but no data is committed to the database. However, the diagnostic data is always written to GKARLOG (as an autonomous transaction). You can use AUDIT mode to test your rule sets without updating the database. UPDATE The rule set will be executed and data will be committed using the value defined in the **Execute Mode** field on GKVPRST.
	Exception Mode	Yes	Indicates how to manage unhandled exceptions in the process actions.	1 Log the error and abort the rule set 2 Log the error and continue processing 3 Ignore the error (this is no longer supported and will

Parameters	Name	Required?	Description	Values
				be removed in a future release)
	Diagnostic Severity	Yes	Severity of the diagnostic information (progress, information, warning, error).	10 Detailed diagnostic information for debugging purposes 20 Performance Diagnostics 30 Progress Messages 40 Warning / non-fatal exception messages 50 Fatal Exceptions

Forms

The form descriptions are provided in this handbook until they can be incorporated into the Banner® General Online Help.

This chapter describes the following forms:

Activity Set-up (GKAPACT) form

Use this form to enter or display the definition of an activity to be associated with a processing task.

Note: The set-up of an activity that is associated with the source indicator `BASELINE` on GKVPACT cannot be modified. You can use the copy functionality on the Generated SQL tab to duplicate the set-up of a baseline activity into a local one, which you can then modify.

Key block

Use this block to specify the activity.

Field	Description
Activity Code	Activity code whose definition you want to enter or display.
Process Code	Process code associated with the activity. Display only.

Source Tables tab

Use this tab to enter or display the source tables used for the activity specified in the Key block. The source tables can be any table that you can use within a SQL statement, such as the following.

- Banner tables
- Microsoft Access tables
- MySQL tables
- Non-Banner Oracle tables
- Text data files that have been registered on the database as external tables (via the Execution Tree Set-up Form (GKAPEXS))

All selected sources must first be defined on the Activity Source Synonym Validation Form (GKVPSYN).

Field	Description
Sequence Number	Sequence number used to specify the order of the source tables/views for this activity. Column `GKRPASO_SEQNO`
Synonym of Source/Table View	Synonym of a source or table view to be used for this activity. Column `GKRPASO_SYNONYMNAME`

Field	Description
Activity Date	Date on which the record was last updated. This field is display only. Column GKRPASO_ACTIVITY_DATE
User	ID of the user who last updated the record. This field is display only. Column GKRPASO_USER_ID

Target Table tab

Use this tab to specify the name and description of the target table for the activity specified in the Key block. You can enter a new name so that a new table will be created, or you can enter or select an existing one.

Field	Description
Target Table Name	Name of the target table. Although you can enter a new name, it is advised that you select an existing one from the list of values when specifying an existing table. This ensures that the associated table comments will be defined on the form as it is on the target table. A warning is displayed when a table comment differs from the one of the existing target table, or when different activities have different table comments for the same target table. Column GKRPATT_TABLE_NAME
Table Description	Free-form text description of the target table. Column GKRPATT_TABLE_DESC
Activity Date	Date on which the record was last updated. This field is display only. Column GKRPATT_ACTIVITY_DATE
User	ID of the user who last updated the record. This field is display only. Column GKRPATT_USER_ID

Target Columns tab

Use this tab to define the columns to be created or used in the target table.

Note: You cannot use this form to change baseline tables, but you can change the specifications for columns of generated Mass Data Update Utility tables.

Field	Description
Sequence Number	Sequence number used to specify the order of the columns for this table. Column `GKRPATC_SEQNO`
Column Name	Name of the column in the target table. Although you can enter a new name, it is advised that you select an existing one from the list of values when specifying an existing column. This ensures that the associated column specifications (datatype, length and scale, nullable indicator and column comment) will be defined on the form as they are on the existing target table. A warning is displayed when a column is defined in several different ways (for example, on different activities or when the definition of a column on an activity differs from the definition of the existing column in the database). Column `GKRPATC_COLUMN_NAME`
Column Description	Free-form text description of the column. Column `GKRPATC_COLUMN_DESC`
Data Type	Data type for the column. Column `GKRPATC_COLUMN_DATA_TYPE`
Column Length	Maximum number of characters that can be entered in the column. A value cannot be entered in this field if the value in the **Data Type** field is `Date` or `CLOB`. Column `GKRPATC_COLUMN_LENGTHPRECISION`
Scale	Number of characters in the column that will be placed after the decimal point. A value can be entered in this field only if the value in the **Data Type** field is `Number`. Column `GKRPATC_COLUMN_NUMBERSCALE`

Field	Description
Nullable	Indicates whether the value in this column can be null. • Selected = The value in this column can be null • Cleared = The value in this column cannot be null Column `GKRPATC_COLUMN_NULLABLE`
Activity Date	Date on which the record was last updated. This field is display only. Column `GKRPATC_ACTIVITY_DATE`
User	ID of the user who last updated the record. This field is display only. Column `GKRPATC_USER_ID`

Select Parts tab

Use this tab to specify the source column or SQL expression that will be used to populate data into the target table's columns. You can use a PL/SQL function to manipulate data, for example, you could use `substr(termcodecolumn,1,4)` to use only the year from a term code.

Field	Description
Sequence Number	Sequence number used to specify the order of the columns for this table. Column `GKRPATS_SEQNO`
Target Column	Name of the column, as defined on the Target Columns tab, in the target table. Column `TARGET_COLUMN`
Select SQL	SQL expression to be used to populate the column specified in the **Target Column** field. Column `GKRPATS_SOURCESELECT`
Populate Key	Indicates whether this column takes part in the comparison with existing data in the target table, before a modification takes place. The set of target columns for which the **Populate Key** check box is selected represents a virtual unique key on the target table for all data inserted through the Mass Data Update Utility.

Field	Description
	If all values for all the columns for which the **Populate Key** check box is selected exist on the target record, then the record is updated with the new values, otherwise a new record is inserted. Use this check box to prevent the creation of duplicate key records. • Selected = This column is part of the virtual key on the target table • Cleared = This column is not part of the virtual key on the target table Column `GKRPATS_ISKEY`
Purge Key	Indicates whether this column takes part in the comparison with existing data in the target table, before a purge takes place. If the purge rule is executed, only records that match values from the rule for all columns with the **Purge Key** check box selected will be deleted. • Selected = This column is used to determine if a record must be deleted by a purge rule. • Cleared = This column is not used to determine if a record must be deleted by a purge rule. Column `GKRPATS_ISKEYFORPURGE`
Activity Date	Date on which the record was last updated. This field is display only. Column `GKRPATS_ACTIVITY_DATE`
User	ID of the user who last updated the record. This field is display only. Column `GKRPATS_USER_ID`

Join/Filter tab

Use this tab to define joins and filters for the source tables used for the activity specified in the Key block. If desired, you can use a SQL development tool to write the SQL expressions, then copy and paste into the **Join** field.

The SQL generated through this tab is displayed on the Generated SQL tab. Comments in the SQL rules should either be enclosed in `/*<comment>*/` or followed by a line break when the comment is prefixed by the symbols “--" (minus, minus) to avoid having parts of the automatically added SQL expressions be considered as a comment.

Because the filter condition is not enclosed in parentheses, it can also be used to incorporate final expressions like `GROUP BY` or `ORDER BY`.

The purge rule is generated in such a way that it can have a twofold function.

- If one or more columns with the **Purge Key** check box selected on the Select Parts tab have a parameter (that is, start with the symbol ":") as the value for the **Select SQL** field, then the join and filter conditions are ignored for the generated purge rule. This allows all the records that match those parameters to be deleted from the target table before the execution of the populate rule. For example, if one target column represents a cohort code, the select SQL value for this column is `:COHORT`, and this column is the only column with the **Purge Key** check box selected, then the generated purge rule will delete from the target table all records having as cohort code the value entered for the execution parameter `:COHORT`.
- If at least one column with the **Purge Key** check box selected has a value that is not a parameter, then the join and filter conditions are included in the generated purge rule. This allows a set of records to be deleted that would coincide on the values of the columns with the **Purge Key** check box selected, under the conditions defined as join and filter, before having a new instance of (part of) the record being inserted by the populate rule of the same activity. The most common use of this feature is to purge entire records (all the columns) before a populate rule inserts data (which matches the deleted records on the key columns) in only some of the columns of the target table. Without the use of the purge rule, the populate rule would update some of the columns of the target table, leading to records that might not represent a coherent set of data (for example, records composed of information pieces merged from different rule executions).

This two-fold functionality allows you to do any of the following:

- Create purge rules having either of the above functions
- Have no purge rule (if no column has the **Purge Key** check box selected)
- Deactivate a purge rule (globally, for all users, using the GKAPEXS form, or locally, for a specific user, using the GKAPEXE form)

This offers a flexible framework for you to control the state of a target table before executing an activity populate rule.

Field	Description
Join	SQL expression that is part of the `WHERE` clause applied on the selection of data from the sources. The join is meant to represent the conditions that link the different sources together. In the final generated rule, the join expression is enclosed in parentheses. Column `GKRPAJF_JOIN`
Filter	SQL expression that is part of the `WHERE` clause applied on the selection of data from the sources. The filter is meant to represent the conditions that are applied to restrict the number of records selected from the sources. In the final generated rule, the join expression is

Field	Description
	not enclosed in parentheses. It is automatically prefixed with the operator `AND`. Column `GKRPAJF_FILTER`
Select Distinct	Indicates whether only distinct records should be selected to populate the target table. This option can be used to limit the frequency with which target records are updated and to improve the performance of the rule execution. • Selected = Only distinct records are to be selected • Cleared = All records are to be selected Column `GKRPAJF_SELECTDISTINCT`
Activity Date	Date on which the record was last updated. This field is display only. Column `GKRPAJF_ACTIVITY_DATE`
User	ID of the user who last updated the record. This field is display only. Column `GKRPAJF_USER_ID`

Generated SQL tab

Use this block to take the SQL constructed from the information gathered on the other tabs of the form and either validate the target table definition by comparing it with the actual table in the database, when this table exists, or generate the target table when it does not yet exist.

If the target table is a generated table, the system also attempts to synchronize the actual table with the definition. Table and column comments of generated tables can always be synchronized. You can add new columns to a generated target table, but the modification of a column datatype, length, scale, or nullable indicator is possible only if the target table is a generated table and the column is empty (that is, if the column contains only NULL values or the table is empty).

If a SQL statement does not compile successfully, you can review and troubleshoot the generated code by copying and pasting it into a PL/SQL tool or a text editor.

Field	Description
Compile Activity block Use this block to generate and compile the SQL statements built from the data, as defined on the other tabs, for the activity. Generating the SQL statements associated with an activity includes verifying and synchronizing (when the target table is a generated table) the target table with the specifications of the activity.	

Field	Description
Populate SQL	SQL statement to be used to select data from the source table(s) to populate the target table. Display only. Column GKVPACT_POPULATESQL
Populate Message	Log of the generation and compilation of the SQL statement associated to the populate action. Display only. Column GKVPACT_POPULATEMESSAGE
Purge SQL	SQL statement to be used to select data to be purged from the target table. Display only. Column GKVPACT_DELETESQL
Delete Message	Log of the generation and compilation of the SQL statement associated to the purge action. Display only. Column GKVPACT_DELETEMESSAGE
Compile Activity	Select this button to compile the SQL statement(s) associated with the selected activity.
Copy/Purge Activity block Use this block to delete all information associated to a chosen Activity on form GKAPACT or to copy the definition of an activity into another activity.	
Source Activity	Code of the activity whose definition you want to copy or purge. This does not need to be the activity specified in the Key block, but the target activity code must exist on GKVPACT. Column SOURCE_ACTI_CODE
Target Activity	Code of the activity to which the source activity's definition is to be copied. This does not need to be the activity specified in the Key block. Column TARGET_ACTI_CODE
Result (untitled)	Result of the copy or purge activity, displayed after the **Copy Activity** or **Purge Source Activity** button is selected. Column GENERATE_COPY_ACTI_MESSAGE

Field	Description
Copy Activity	Select this button to copy the source activity's definition to the target activity.
Purge Source Activity	Select this button to delete all GKAPACT definitions associated with the activity code specified in the **Source Activity** field. The activity code itself is not deleted on GKVPACT.

Process Tree Execution (GKAPEXE) form

Use this form to set up and execute one or more stages of a processing task. You can also use this form to export data from a table into an XML file.

Key block

Use this block to specify the process for which you want to set up and execute one or more stages.

Field	Description
Process Code	Process code.

Execution Tree Set-up tab

Use this tab to specify the stage(s) to be executed. Changes made on this tab are made to only the current instance (that is, private to the current user) of the process and do not change details on the Execution Tree Set-up Form (GKAPEXS).

An activity can be executed only if no user is currently executing it. The **Compile Tree** button in the main window and the **Run From Jobsub** and **Run Now** buttons on the Execution tab verify this for each activity before executing it.

Note: The default export directory name is SCHEMA_XMLDIR and path is $BANNER_HOME/mduu/xml. The users can add their own directories with the help of DBAs and use them as well.

Field	Description
Tree Pane (untitled)	Process's "tree" structure and the execution status of each node (process, tasks, and activities). Column `PTATREE`

Field	Description
Node	Code of the node selected in the tree pane. Display only. Column GKRPPTR_CHILD
Status	Status of the node for this instance. This does not necessarily match the status of the task or activity on GKAPEXS. Also, status changes made on this form do not change the status on GKAPEXS. Column GKRPPTR_CURRENT_STATUS
Result (untitled)	Result of a status change or tree compilation, displayed after the **Change Status** or **Compile Tree** button is selected. Column NODE_MESSAGE
Start Date	Date and time on which the selected node was last started for the current user. Display only. This field is not displayed if a task or the root process node is selected in the tree pane. Column GKRPPTR_STARTED
End Date	Date and time on which the selected node was last ended for the current user. Display only. This field is not displayed if a task or the root process node is selected in the tree pane. A NULL value when the **Start Date** field is not NULL indicates that the current user has started the execution of the activity and that this has not yet been completed. Column GKRPPTR_FINISHED
Compile Tree button	Select this button to compile the tree. The system uses only the tasks and activities whose status is Enabled or Enabled with purge.
Change Status button	Select this button to change the status of the item selected in the tree pane. The status change will be reflected in the tree pane and in the **Status** field.
Mark All Nodes as Not Running button	Select this button to reset the **End Date** field value after a failure. In some cases, such as after a loss of database connection during an execution, the **End Date** field is not updated when the execution is completed. This button

Field	Description
	allows you to mark, for the current user, all activities as not running. This button updates the end date NULL values *without* verifying whether the activities are effectively running or not.

Execution tab

This tab is used to view and update the parameters that are used by the process code (as defined on the Business Rules Process Parameters Form (GKRSQPA)).

When the rule set is completed successfully, a pop-up window will show the process sequence number and confirms the Process and Rule Set Codes.

If an unhandled exception is encountered causing the rule set to fail, then an error message is displayed showing the sequence number.

The diagnostic data is stored on the Diagnostic Form (GKARLOG).You can use the View SQL Log option to navigate to GKARLOG.

Field	Description
Execution Mode	Execution mode is used by the delivered actions. Locally developed actions can make use of this to meet specific institutional requirements. `AUDIT` (default) The rule set is executed but no data is committed to the database. However, the diagnostic data is always written to GKARLOG (as an autonomous transaction). You can use AUDIT mode to test your rule sets without updating the database. `UPDATE` The rule set will be executed and data will be committed using the `RULE` mode (specified in the **Execute Mode** field on GKVPRST). Column `EXECUTION_MODE`
Exception Mode	Indicates how to manage unhandled exceptions in the process actions. • Log the error and abort the rule set • Log the error and continue processing • Ignore the error (this is no longer supported and will be removed from the form in a future release)

Field	Description
	Column EXCEPTION_MODE
Diagnostic Severity	Severity of the diagnostic information (progress, information, warning, error). Indicates what information will be displayed in the error log. The convention for the severity number is as follows: 1 Detailed diagnostic information for debugging purposes 20 Performance Diagnostics 30 Progress Messages Column DIAGNOSTIC_SEVERITY
Parameter	Parameters that are used by the process code and defined on GKRSQPA. Display only. Column PARAM_SQPA_DESC
Parameter Value	Value assigned to the parameter for the process run. Column PARAM_DEFAULT_VALUE
Run From Jobsub button	Select this button to run the Business Rule Process (GKPPSQL) from the GJAPCTL form. **Note:** Dynamic parameters are built into a ruleset when GKPPSQL launches GJAPCTL using the Run From Jobsub option on GKAPEXE or GKAPPLN. An error can occur when attempting to execute GKPPSQL directly from GJAPCTL without first building the ruleset on GKAPEXE or GKAPPLN. A one-up sequence number is generated to identify the parameter set. The parameters and default values will populate the GJAPCTL form, including the Parameter Set number (that is, the one-up sequence number that is generated in GKAPEXE). By submitting the process through GJAPCTL, baseline functionality is available.
Run Now button	Select this button to initiate the Business Rule Process (GKPPSQL).

Field	Description
	Note: Dynamic parameters are built into a ruleset when GKPPSQL launches GJAPCTL using the Run From Jobsub option on GKAPEXE or GKAPPLN. An error can occur when attempting to execute GKPPSQL directly from GJAPCTL without first building the ruleset on GKAPEXE or GKAPPLN. A one-up sequence number is generated and a pop-up-window displays the number. A job sequence number is not generated. While the process is running, the Banner session will be engaged. The form will display a message when the processing is complete.

Export tab

Use this tab to export a table or an activity. The table or activity does not need to be associated with the process specified in the Key block. Exported structures can then be re-imported on the Execution Tree Set-up Form (GKAPEXS), on the same database, or another one (for instance, to migrate set-up data from pre-production to production).

Field	Description
Export Table block Use this block to export table data to an XML file to be created with the name and in the directory you specify.	
Table Name	Name of the table to be exported. Column `SAVETABLE_TABLENAME`
Directory Name	Name of the directory to which the table is to be exported. Column `SAVETABLE_DIR_CODE`
Directory Path (untitled)	Path of the directory to which the table is to be exported. Display only. Column `SAVETABLE_PATH`
File Name	Name of the file to which the table data is to be exported. Column `SAVETABLE_FILENAME`

Field	Description
Conditions	Condition(s) for selecting records to be exported. This option allows you to export a subset of the table data. Column SAVETABLE_WHERE
Result (untitled)	Result of the table export, displayed after the **Export Table** button is selected. Column SAVETABLE_MESSAGEOUT
Export Table button	Select this button to export the table structure (table name and comment, column names, and definitions) and data. The system creates an XML file containing the contents of the export in the specified directory and opens a new browser window to display the contents of the XML file.
Export Activity block Use this block to export the definition of an activity. You can use this block, for example, to export an activity from a development database to a production database.	
Activity Name	Name of the activity to be exported. Column SAVEACT_ACTI_CODE
Activity Description (untitled)	Description associated with the activity code. Display only. Column ACTI_CODE_DESC
Directory Name	Name of the directory to which the activity is to be exported. Column SAVEACT_DIR
Directory Path (untitled)	Path of the directory to which the activity is to be exported. Display only. Column SAVEACT_PATH
File Name	Name of the file to which the activity data is to be exported. Column SAVEACT_FILENAME
Result (untitled)	Result of the activity export, displayed after the **Export Activity** button is selected. Column SAVETABLE_MESSAGEOUT
Export Activity button	Select this button to export the activity. The system creates an XML file containing the contents of the export in the specified directory.

Field	Description
	The XML file is displayed in a new browser window.

Execution Tree Set-up (GKAPEXS) form

Use this form to define or display the execution tree for a processing task. You can also use this form to import data from a table.

Key block

Use this block to specify the process for which you want to define or display the execution tree.

Field	Description
Process Code	Process code.

Tree Set-up tab

Use this tab to define the execution tree for the process specified in the Key block. The structure of an execution is a root (the process code), of which there can only be one; many children (tasks) that can be associated with the root; and many grandchildren (activities) that can be associated with a task.

The diagram on page 9 is an example of such a tree.

Activities cannot be assigned to a root; they can only be assigned to a task.

Note: **Known Limitation:** Although it is possible to repeat a task under a chosen process root, the generated rule set will not respect the chosen order of the task instances. Therefore, if a task needs to be run several times for a chosen process, it is strongly advised not to use the same task twice on the tree, but rather to create another task and associate to it the same activities as on the initial task.

To add a task to the root, or to add an activity to a task, select the **Select to Insert Node Here** item in the tree pane in the appropriate place, then select the task or activity in the **Node Code** field. The system will create a new item in the structure, which you can then define in the right pane.

Field	Description
Tree Pane (untitled)	Process's execution tree structure and the execution status of each task and activity. Column `PTATREE`

Field	Description
Node	Code of the task or activity selected in the tree pane. If the **Select to Insert Node Here** item is selected, you can enter or select a node code. If any other item is selected, this field is display only. The values included on the list depend on the depth of the selected node. When a child of the root is selected, the list displayed contains the tasks associated to the root process code (GKVPTAS). When a grandchild of the root is selected, the list displayed contains the activities associated to the root process code (GKVPACT). Column `GKRPPTR_CHILD`
Status	Status of the task or activity for execution. Use the **Change Status** button to change the status. Column `GKRPPTR_CURRENT_STATUS`
Result (untitled)	Result of a status validation, displayed after the **Validate Status** button is selected. Column `VALIDATE_MESSAGE`
Populate SQL	SQL statement to be used to select data from the source table(s) to populate the target table. Display only. This field is not displayed if the process root or a task is selected in the tree pane. Column `GKVPACT_POPULATESQL`
Delete SQL	SQL statement to be used to purge the target table. Display only. This field is not displayed if the process root or a task is selected in the tree pane. Column `GKVPACT_DELETESQL`
Change Status button	Select this button to change the status of the item selected in the tree pane. The status change will be reflected in the tree pane and in the **Status** field.
Validate Status button	Select this button to validate the current status of a node. For example, an enabled activity might have been invalidated by a change on GKAPACT or by a database change. In this

Field	Description
	case, the validation will fail and the status of the activity will be updated to disabled.

Import tab

Use this tab to register a file as an external table or to import an XML file generated on the Process Tree Execution Form (GKAPEXE). The table or file does not need to be associated with the process specified in the Key block.

External tables for registered data files are always created in the private schema associated with the Mass Data Update Utility.

Field	Description
External File Registration block Use this block to create an external table associated with a file on the database server (or accessible to it through mapped drives). The file can contain any type of data supported by Oracle for external tables including .csv and fixed length text files.	
External Table Name	Name of the external table to be imported created and associated to the file. Column `REGISTERFILE_TABLENAME`
Directory Name	Name of the directory in which the file to be registered resides. Column `REGISTERFILE_DIR`
Directory Path (untitled)	Path of the directory which the table to be imported resides. Display only. Column `REGISTERFILE_PATH`
File Name	Name of the file to be registered. Column `REGISTERFILE_FILENAME`
Record Delimiter	Delimiter to be used to mark the end of a record in the file. The usual and default value is `NEWLINE`. (Refer to the Oracle RDBMS documentation on external tables for more information). Column `REGISTERFILE_RDELIM`
Fields	Comma delimited list of the column name and size to be created. The syntax of a list item is either `columnname(size)` for a

Field	Description
	symbol-delimited data file or `columnname(startingposition:length)` for a fixed-length data file. For example, if the field delimiter value is, (comma), then `A(10), B(20)` would result in two columns: column A 10 characters in length and column B 20 characters in length. If the field delimiter value is `FIXEDLENGTH`, then `A(1:10), B(100:20)` would result in two columns: column A for the 10 characters starting from position 1 and column B for the 20 characters starting at position 100 (on each record in the data file). Column `REGISTERFILE_FIELDS`
Field Delimiter	Delimiter to be used for fields. Usual values are `FIXEDLENGTH` and, (comma). Column `REGISTERFILE_FDELIM`
Drop the external table if it exists	Indicator used to specify whether an existing version of the table, specified in the **External Table Name** field is to be dropped before it is recreated. Only tables in the private schema of the Mass Data Update Utility (that is, tables generated from the Mass Data Update Utility) can be dropped. Column `REGISTERFILE_DROP`
Result (untitled)	Result of the external table registration, displayed after the **Register File** button is selected. Column `REGISTERFILE_MESSAGEOUT`
Register File button	Select this button to register the external table associated with the chosen data file.
XML-Extract Import block (untitled) Use this block to import an XML file that was created on the Process Tree Execution Form (GKAPEXE).	
Directory Name	Name of the directory in which the XML file to be imported resides. Column `IMPORTXML_DIR`
Directory Path (untitled)	Path of the directory in which the XML file to be imported resides. Display only.

Field	Description
	Column IMPORTXML_PATH
File Name	Name of the file to be imported. Column IMPORTXML_FILENAME
Verbose	Indicates whether the log of the import should be returned in a detailed form. Column IMPORTXML_VERBOSE
Result (untitled)	Result of the file import, displayed after the **Import XML File** button is selected. Column IMPORTXML_MESSAGEOUT
Import XML File button	Select this button to import the selected XML file.

Note: The default import directory name is SCHEMA_XMLDIR and path is $BANNER_HOME/mduu/xml. The users can add their own directories with the help of DBAs and use them as well.

Process Transaction Maintenance (GKAPMLT) form

Use this form to view, add and update data in the Process Transactions table (GKRPWRK). The data is usually the result of executing a rule set action from the GKAPPLN form.

You can change the data to suit your needs, and then extract the data to the user's desktop using a modified version of the Help Data Extract functionality.

Note: If you changed any data, you may want to bring that changed data back into Banner. You will need to create another rule set to bring the data back into Banner.

Some restrictions apply to GKAPMLT.

- If the GKRWRK_PIDM field is selected to be visible, it will always appear as the left-most field on the form and will always be displayed with the **Name** field to its right.

  It is not possible to query or order by the **Name** field attached to the **ID** field.

  Users may work around these restrictions by populating any of the generic columns with name, ID or PIDM.
- Previous Block cannot be used to navigate from the Process Transactions block to the Filter block. Instead, you must Rollback to the Key block.
- The baseline Help Data Extract functionality is not supported on this form.
- Data generated with previous versions of the Mass Data Update Utility are not guaranteed to work with this version.
- The following columns are not currently supported in the GKAPMLT form:

- GKRWPRK_V51 through to GKRPWRK_V99
- GKRPWRK_B01
- GKRPWRK_L11 through to GKRPWRK_L99

Key block

Use this block to specify the process, action, rule, and rule set for which you want to enter or display GKRPWRK data.

Field	Description
Process Code	Process code. Column SQPR_CODE
Action	Action code. Column SVBA_CODE
Rule	Rule code. Column SQRU_CODE
Rule Set	Rule set code. Column PRST_CODE
Summary	Select this button to display the Process Transaction Inquiry Form (GKIPRTR).

Filter block

This block is used to enter a query value for the auto-populate code. The query value is used in the query that produces the results that are viewed in the Process Transaction block.

This block is populated from the GKRPRCT rows that are associated with selections in the Key block.

Field	Description
Auto-Populate Description	Description of the database column that will be automatically populated. Display only. The description is derived from GKRPRCT by getting the rule for the column where the process code, rule code, action code and column match. GKRPRCT_SVAP_CODE contains a column name; the description for this column is contained in GKRPRCT_SVAP_DESC (if the column is null, then the description is derived

Field	Description
	from GKVSVAP_DESC; if this column is null, then GKVPRCT_SVAP_ABBR is used; if this column is null, then the value in GKVPRCT_SVAP_CODE is used - i.e., the GKRPWRK column name). Column GKRPRCT_SVAP_DESC
Query Value	Value for the auto-populate code to use in the selection criteria. This field is used to filter the data that is retrieved into the Process Transactions block. Only data that contains the entered value for the column will be displayed on the Process Transactions block. Column GKRPRCT_QUERY_VALUE
Include Null	Indicates that a record should be display even if the value for the auto-populate code is null. • Selected = Display the record even if the auto-populate code is null • Cleared = Do not display the record if the auto-populate code is null Column GKRPRCT_INCLUDE_NULL_IND
Display	Allows users to hide or show columns on the Process Transaction block.
Order Priority	Indicates the priority (lowest number = highest priority) by which the data will be ordered in the Process Transaction block. This field is populated with the data from the **Order Sequence** field on the GKRPRCT form (GKRPRCT_ORDER_SEQNO) for matching process code/action/rule combinations. The value can be changed.
Order By	Indicates if the rows should be sorted by ascending or descending values. This field is populated with the data from the **Order By** field on the GKRPRCT form (GKRPRCT_ASCDESC) for matching process code/action/rule combinations. The value can be changed.

Process Transaction block

This block displays dynamic column headings, which have variable widths based on the GKRPRCT rules for the column. Open this block by selecting the Working Results option.

The rows are sorted by ascending or descending values based on the value entered in the **Order By** field on the Filter block.

The data is from the GKRPWRK table.

Field	Description
ID	Student ID. The ID is displayed when the **Display** check box is selected on the Filter block for the ID. Display only. This field is displayed when `GKRPWRK_PIDM` is entered in the **Auto-Populate Code** field, and values are entered in the **Abbreviation**, **Display Sequence** and **Order Sequence** fields on the GKRPRCT form. The ID field will always be displayed first, regardless of the value entered in the **Display Sequence** field. Records can be queried in this field. The ID is derived from `GKRPWRK_PIDM` and the SPRIDEN record.
Name	Student name. The ID and name are displayed when the **Display** check box is selected on the Filter block for the ID. Display only.
Variable Columns	These fields have varying lengths and titles. The Business Rules Auto-Populate Rules Form (GKRPRCT) is used to determine which fields are displayed, how they are displayed and whether they can be updated. The auto-hint that is displayed for a field is from the **Auto-Populate Description** field on the GKRPRCT form. An LOV (List of Values) can be associated with a column, by entering a validation table in the **Entity** field on the GKRPRCT form.
Extract Data	Extracts the data displayed on the form to the user's desktop using a modified version of the Help Data Extract functionality.

Field	Description
	Data that is extracted is sorted by ascending or descending values based on the value entered in the **Order By** field on the Filter block. This data extract functionality does not support the WEBUTIL method for extracting and saving data. The `Internet Native data extract format` setting, in the Directory Options tab on the General User Preferences Maintenance form (GUAUPRF), determines if a text or comma-separated file is created. The data extract function operates as follows: • Each row has a size limit of 4,000 characters (baseline limit is 2,000). • Trailing spaces are removed from each field. • Column headings are extracted from the definitions in the Display Options tab of GKRPRCT. • Carriage returns will not be extracted. • Data and header text can contain a single quote (') or double quote (").

Process Launch (GKAPPLN) form

Use this form to specify the parameter values and execute the rule set. Executing a rule set will run the SQL rules (where the SQL is associated with the rule code on the GKRRSQL), and will carry out the action(s) against each row of data returned.

The rules and actions are assigned to a process/rule set combination from the GKRPRST form.

The GKPPSQL process is used to execute the process/rule set combination. The GKPPSQL process can be run directly from this form, or it can be run from the GJAPCTL form.

Note: Dynamic parameters are built into a ruleset when GKPPSQL launches GJAPCTL using the Run From Jobsub option on GKAPEXE or GKAPPLN. An error can occur when attempting to execute GKPPSQL directly from GJAPCTL without first building the ruleset on GKAPEXE or GKAPPLN.

Users will execute a business rule or a set of business rules, which can be used, for example, to group students, and store values that will identify the students in a specific group to make recommendations or to update student records.

Key block

Field	Description
Process Code	Process code for which the rule set will be executed. Users can only select process codes that they are allowed to use based on the information entered on the Process Rules Roles Form (GKAPRRO). Column `GKRPRST_SQPR_CODE`
Rule Set	Rule set code for which the rule set will be executed. Users can only select process codes that they are allowed to use based on the information entered on the Process Rules Roles Form (GKAPRRO). Users have access to a rule set if they have access to the process code, and to all of the active/valid/date-effective rules and actions in the rule set. Column `GKRPRST_PRST_CODE`
Effective Date	Effective date that will be compared to the rules. The current date is the default value. Display only. When the user selects Next Block, the value in **Effective Date** field is compared to the rule start and end dates, and the parameter start and end dates.

Process Launch Parameters tab

This tab is used to view and update the parameters that are used by the process code/rule set combination.

About this task

The rule and action parameters are both displayed on this block. If a parameter is used multiple times within the rule set and different default values are used, then this block displays a null parameter value. The user will need to assign a value if the parameter value is null.

The rule parameters that are displayed are selected as follows:

Procedure

1. The rule parameters with default values are retrieved from the GKRPRSP table (Process Rule Set Parameters block on GKRPRST).
2. If no records exist in GKRPRSP, then the rule parameters and default values are retrieved from the GKRSQRP table.
3. Parameters that exist in the GKRSQRP table but are missing from the GKRPRSP table are retrieved from GKRSQRP.
4. If no records exist in the GKRPRSP and GKRSQRP tables, then the parameters are retrieved from the GKRSQPA table for the process code (rather than the SQL rule code).

   The action parameters that are displayed are selected as follows:
5. The action parameters with default values are retrieved from the GKRPACP table (Action Parameters block on GKRPRST).
6. If no records exist in GKRPACP, then the action parameters and default values are retrieved from the GKRSVBA table.

Results

When the rule set is completed successfully, a pop-up window will show the process sequence number (under which the diagnostic data is stored in GKARLOG) and confirms the Process and Rule Set Codes.

If an unhandled exception is encountered causing the rule set to fail, then an error message is displayed showing the sequence number.

Field	Description
Execution Mode	Execution mode is used by the delivered actions. Locally developed actions can make use of this to meet specific institutional requirements. `AUDIT` (default) The rule set is executed but no data is committed to the database. However, the diagnostic data is always written to GKARLOG (as an autonomous transaction). You can use AUDIT mode to test your rule sets without updating the database. `UPDATE` The rule set will be executed and data will be committed using the value defined in the **Execute Mode** field on GKVPRST.
Exception Mode	Indicates how to manage unhandled exceptions in the process actions. • Log the error and abort the rule set • Log the error and continue processing

Field	Description
	• Ignore the error (this is no longer supported and will be removed from the form in a future release)
Diagnostic Severity	Severity of the diagnostic information (progress, information, warning, error). Indicates what information will be displayed in the error log. The convention for the severity number is as follows: 10 Detailed diagnostic information for debugging purposes 20 Performance Diagnostics 30 Progress Messages 40 Warning / non-fatal exception messages 50 Fatal Exceptions
Parameter	Parameters that are used by the process code/ rule set/action code combination. The rule and action parameters are both displayed. Display only.
Parameter Value	Default value of the parameter. If a parameter is used multiple times within the rule set and different default values are used, then this block displays a null parameter value. A value needs to be assigned if the parameter value is null.
Run From Jobsub	Opens the GJAPCTL form with the Business Rule Process (GKPPSQL). **Note:** Dynamic parameters are built into a ruleset when GKPPSQL launches GJAPCTL using the Run From Jobsub option on GKAPEXE or GKAPPLN. An error can occur when attempting to execute GKPPSQL directly from GJAPCTL without first building the ruleset on GKAPEXE or GKAPPLN. one-up sequence number is generated to identify the parameter set. The parameters and default values will populate the GJAPCTL form, including the Parameter Set number (that is, the one-up sequence number that is generated in GKAPPLN). By submitting the process through GJAPCTL, baseline functionality is available.

Field	Description
Run Now	Initiates the Business Rule Process (GKPPSQL), which will run in the background. **Note:** Dynamic parameters are built into a ruleset when GKPPSQL launches GJAPCTL using the Run From Jobsub option on GKAPEXE or GKAPPLN. An error can occur when attempting to execute GKPPSQL directly from GJAPCTL without first building the ruleset on GKAPEXE or GKAPPLN. A one-up sequence number is generated and a pop-up-window displays the number. A job sequence number is not generated. While the process is running, the Banner session will be engaged. The form will display a message when the processing is complete.

Process Rule Details tab

Use this tab to view the rules associated with the selected process code and rule set. This tab is display only.

Field	Description
Process Rule Set block This block displays the rule code information for selected process code and rule set. The information is from the Business Process Rule Set Form (GKRPRST). This block is display only.	
Run Sequence	Sequence number, which indicates the order in which the rule will be run. Display only. Column `GKRPRST_SEQ_NO`
Rule	Rule code for the SQL statement. Column `GKRPRST_SQRU_CODE`
Rule Description	Description of the rule code. Display only.
Active	Indicates if the rule is active. Display only. The value can only be overridden from GKRPRST. • Selected = Rule is active • Cleared = Rule is not active Column `GKRPRST_ACTIVE_IND`

Field	Description
Effective Rules	Indicates the number of SQL statements for the rule (GKRPRST_SQRU_CODE) that are active as of the effective date from the Key block. The count is derived from GKRRSQL. Display only.
Total Rules	Indicates the total number of SQL statements for the rule (GKRPRST_SQRU_CODE). The count is derived from GKRRSQL. Display only.
Process Rule Set Actions block This block displays the rule set actions for the selected process code and rule set. The information is from the Rule Set Actions block on the Business Process Rule Set Form (GKRPRST). This block is display only.	
Seq (unlabeled)	Order in which the action(s) will be performed.
Action	Action code that determines what the set of SQL statements will do. The value is from GKRPACT_SVBA_CODE.
Active	Indicates if the rule is active. Display only. The value can only be overridden from GKRPRST. • Selected = Rule is active • Cleared = Rule is not active The value is from GKRPACT_ACTIVE_IND.
Function	Database function name. Display only. The value is from GKVSVBA_DBPROC_NAME.
Start Date	Action start date. Display only. The value is from GKVSVBA_START_DATE.
End Date	Action end date. Display only. The value is from GKVSVBA_END_DATE.
Execute Ruleset	Accesses the Process Launch Parameter tab.

Universal Viewer (GKAPUNV) form

Use this form to view and maintain the data in an institution-defined table. Only tables owned by the PRGNREP schema can be viewed in this form.

Note: The PRGNREP schema does not enforce Banner table naming conventions, so tables can have more or fewer than seven characters, and they will be viewable in this form.

You can add, modify, and delete records in these user-defined tables as long as they are not Oracle external tables. You can also extract data, either the entire table or the data in a single field.

If desired, you can reduce the number of records displayed by specifying a value for a column, up to three columns, using the filter fields in the Key block. Queries can also be performed as usual in the base block.

Take note of the following restrictions and characteristics of the form:

- Data fields for each column in the table are generated automatically using a fixed length field.
- Only the first 200 columns of each table will be displayed.
- Large objects (CLOB, BLOB) are truncated to the first 100 characters and cannot be updated on the form.
- The baseline Help>Data Extract functionality can be used but will not extract CLOB or BLOB data.
- The form does not support the display or update of data in Oracle external tables.

Key block

Use this block to specify the table to be viewed. You can also filter up to three columns' values, if desired. If you leave the filter fields blank, all records will be displayed when you navigate to the base block.

Field	Description
Table Name	Name of the table whose records you want to view. The table must be owned by the PRGNREP schema.
Filter Column	Name of the column for which you want to filter data. You can filter up to three columns by entering a different column name in each **Filter Column** field.
Filter Value	The value or wildcard to use in the filter.

Base block

This block displays the table in a tabular view. The block is built 'on the fly' for the table specified in the Key block and does not refer to database columns; instead they refer to columns in a dynamic SQL statement based on the table specified in the key block.

Columns are displayed in alphabetical order by column name.

Note: If a column name begins with the table name, the system does not display the first part of the column name. For example, if a table name is ABCDEFG, and a column name is ABCDEFG_ACADEMIC_YEAR, the form displays ACADEMIC_YEAR for the column name.

In this block, you can add, modify, and delete records in these user-defined tables as long as they are not Oracle external tables. Also, you cannot update Oracle large objects.

You can also extract the table data to your browser or a spreadsheet program such as Microsoft Excel using one of the following extract options on the Help Menu.

- Use the Extract Data with Key option to include the data in the Key block and the table data.
- Use the Extract Data No Key option to include only the table data without the data in the Key block.

  Note: The data extract function does not extract the value of any Oracle large objects such as CLOB or BLOB.

You can extract the content of a single field by double clicking in the field. You can extract the content of a column by double clicking in one field of the chosen column. The system displays the Export Data window, which you use to specify the directory and file name.

Export Data window

Use this window to export extract data from a field in the base block at the cell level or the column level. This is particularly useful if a field contains a large amount of data, such as a Character Large Object (CLOB) field containing XML, that cannot be displayed in its entirety.

When you select **Export Cell Data** or **Export Column Data**, the system displays the data in your browser using the application that matches the file extension that you provide; for example, `xxx.doc` would be displayed as a Microsoft Word document. If you specify an extension for the filename that is not know to the local system, the system asks if you want to save the file. If this happens and you want to display the data, rather than save it, cancel the action and enter an appropriate extension in the **Filename** field.

Field	Description
Selected Table	PRGNREP table name specified in the Key block. Display only.
Selected Column	Name of the column selected in the base block. Display only.
Filter	SQL statement of the filter in effect for the export. If you specified column and value filters in the Key block, the SQL filter defaults to this field. You can add more conditions, modify the conditions, or delete either the conditions or the entire filter.
Directory	Oracle directory name. Directories can be created by the database administrator to provide an association to a folder on the database server. Check with your local DBA for details about what is available.

Field	Description
Filename	Name of the file. The filename extension must be correct for the type of data in the field. For example, if the field contains XML, you should use .xml as the extension; if the field contains plain text, you should use .txt as the extension. If you do not provide an extension, the system adds .text by default.
Export Cell Data button	Select this button to export cell data. The system saves the data to the file as specified in the **Directory** and **Filename** fields; it also launches a new browser window and displays the data in that window so you can easily check the results to make sure they are what you wanted.
Export Column Data button	Select this button to export column data. The system saves the data to the file as specified in the **Directory** and **Filename** fields; it also launches a new browser window and displays the data in that window so you can easily check the results to make sure they are what you wanted.
Cancel button	Select this button to cancel the export.

Activity Source Synonym Validation (GKVPSYN) form

Use this form to enter or display synonym names for tables used in activities assigned to processing tasks.

The source tables can be any table that you can use within a SQL statement, such as the following:

- Banner tables
- Microsoft Access tables
- MySQL tables
- Non-Banner Oracle tables
- Text data files that have been registered on the database as external tables (via the Execution Tree Set-up Form [GKAPEXS])

Note: A synonym that is associated with the source `BASELINE` cannot be modified.

Field	Description
Synonym Name	Synonym name associated with a table. Column `GKVPSYN_SYNONYMNAME`

Field	Description
Description	Description of the table. Column GKVPSYN_DESC
Source	Source of the record. Valid values are the following. Baseline The record was a delivered value. Local The record was created locally. Temporary The record was created during an import/export activity and will be purged automatically. Column GKVPSYN_SOURCE_IND
User ID	ID of the user who last updated the record. This field is display only. Column GKVPSYN_USER_ID
Activity Date	Date on which the record was last updated. This field is display only. Column GKVPSYN_ACTIVITY_DATE

Process Task Code Validation (GKVPTAS) form

Use this form to enter or display codes for processing tasks.

Note: An activity that is associated with the source BASELINE cannot be modified. It can be duplicated into a local copy, which you can then modify.

Field	Description
Task Code	Code of the task. Column GKVPTAS_TASK_CODE
Description	Description of the task. Column GKVPTAS_DESC
Process Code	Code of the process with which the activity is associated. Column GKVPTAS_SQPR_CODE
Source	Source of the record. Valid values are the following.

Field	Description
	`Baseline` The record was a delivered value. `Local` The record was created locally. `Temporary` The record was created during an import/export activity and will be purged automatically. Column `GKVPTAS_SOURCE_IND`
User ID	ID of the user who last updated the record. This field is display only. Column `GKVPTAS_USER_ID`
Activity Date	Date on which the record was last updated. This field is display only. Column `GKVPTAS_ACTIVITY_DATE`

Process Rules Roles (GKAPRRO) form

Use this form to create and maintain the Mass Data Update Utility security rules. From this form, you can associate FGAC VBS business profiles with Mass Data Update Utility process codes, action codes and rule codes.

When process codes, action codes and rule codes are associated with FGAC business profiles, only the users assigned to the FGAC business profile will have access to the associated processes, actions and rule codes. Individual users are assigned to business profiles on the FGAC Business Profile Assignment Form (GOAFBPR). In order to have access to a rule set, a user must have access to its process code and all of the rules and actions the rule set contains.

If no active FGAC business profiles are associated to process codes, action codes or rule codes, then no security restrictions will be in place.

Key block

Use this block to enter data that will be used to filter the records that will be listed in the tabs. All of the fields are optional.

Field	Description
Business Profile	FGAC business profile upon which to filter the values listed in the tabs.
Process Code	Process code upon which to filter the values listed in the tabs.

Process Code Profiles tab

This tab is used to associate FGAC business profiles to process codes. If a business profile and process code are associated, then only the users assigned to the business profile will be able to access the associated process code.

If no active FGAC business profiles are associated with process codes, then no security restrictions exist for process codes.

A record cannot be changed, except for the **Active** check box, after a record has been saved. To change a record, you must delete it, and add it again with the new data.

This tab uses the GKRPRRO table.

Field	Description
Business Profile	FGAC business profile to be associated with a process code. The value cannot be changed after the record has been saved.
Process Code	Process code to be associated with a business profile. The value cannot be changed after the record has been saved.
Active	Indicates if security restrictions are active for the associated business profile and process code. • Selected = Active • Cleared = Not active

Action Code Profiles tab

This tab is used to associate FGAC business profiles to process codes and action codes. If a business profile, process code and action code are associated, then only the users assigned to the business profile will be able to access the associated process code/action code combination.

If no active FGAC business profiles are associated with process codes and action codes, then no security restrictions exist for process code/action code combinations.

A record cannot be changed, except for the **Active** check box, after a record has been saved. To change a record, you must delete it, and add it again with the new data.

This tab uses the GKRPRBA table.

Field	Description
Business Profile	FGAC business profile to be associated with a process code and action code. The value cannot be changed after the record has been saved.

Field	Description
Process Code	Process code to be associated with a business profile. The value cannot be changed after the record has been saved.
Action Code	Action code to be associated with a business profile. The value cannot be changed after the record has been saved.
Active	Indicates if security restrictions are active for the associated business profile, process code and action code. • Selected = Active • Cleared = Not active

Rule Code Profiles tab

This tab is used to associate FGAC business profiles to process codes and rule codes. If a business profile, process code and rule code are associated, then only the users assigned to the business profile will be able to access the associated process code/rule code combination.

If no active FGAC business profiles are associated with process codes and rule codes, then no security restrictions exist for process code/rule code combinations.

A record cannot be changed, except for the **Active** check box, after a record has been saved. To change a record, you must delete it, and add it again with the new data.

This tab uses the GKRPRRU table.

Field	Description
Business Profile	FGAC business profile to be associated with a process code and rule code. The value cannot be changed after the record has been saved.
Process Code	Process code to be associated with a business profile. The value cannot be changed after the record has been saved.
Rule Code	Rule code to be associated with a business profile. The value cannot be changed after the record has been saved.
Active	Indicates if security restrictions are active for the associated business profile, process code and rule code. • Selected = Active • Cleared = Not active

Diagnostic (GKARLOG) form

Use this form to view the diagnostic data from when a rule set was run from GKAPPLN.

Key block

The Key block contains the following fields.

Field	Description
Process Code	Process code for which you want to view the diagnostic data. Users can only select process codes that they are allowed to use based on the information entered on the Process Rules Roles Form (GKAPRRO).
Rule Set	Rule set code for which you want to view the diagnostic data. Users have access to a rule set if they have access to the process code, and to all of the active/valid/date-effective rules and actions in the rule set.
Run Sequence	Process sequence number that was assigned to the diagnostic data when a rule set was run from GKAPPLN. When the rule set is completed successfully on GKAPPLN, a pop-up window displays the process sequence number. The default value in this field is the most recent (highest) available sequence for the selected Process Code/Rule Set combination. The validation for the **Run Sequence** field checks that the value in the **User** field matches the value in the **User** field (`GKRRLOG_USER_ID`).
User	User who is logged in to Banner. The value can be changed but Baseline FGAC PII security can be used to mask the field. The purpose of masking the field is to prevent a user from seeing logs generated by other users.

Base block

The Base block contains the following fields.

Field	Description
Sequence	Sequence number of the statement. Column GKRRLOG_STATEMENT_SEQNO
Diagnostic	Description of the diagnostic data. Column GKRRLOG_TEXT
Date	Date and time the data was created. Column GKRRLOG_ACTIVITY_DATE
Severity	Severity of the diagnostic information (progress, information, warning, error). Indicates what information will be displayed. The convention for the severity number is as follows: 10 Detailed diagnostic information for debugging purposes 20 Performance Diagnostics 30 Progress Messages 40 Warning / non-fatal exception messages 50 Fatal Exceptions Column GKRRLOG_SEVERITY

Process Rule Column Definition Inquiry (GKIPRCT) form

Use this form to display the existing column definitions from the GKRPRCT table. The form is not accessible through the navigation menu; it can be accessed through Direct Access or the Business Rules Auto-Populate Rules Form (GKRPRCT).

Field	Description
Process Code	Process code. This field is queryable. Column GKRPRCT_SQPR_CODE
Description	Description associated with the process code. This field is queryable. Column GKVSQPR_DESC

Field	Description
Rule Code	Rule code. This field is queryable. Column GKRPRCT_SQRU_CODE
Description	Description associated with the rule code. This field is queryable. Column GKVSQRU_DESC
Action Code	Action code. This field is queryable. Column GKRPRCT_SVBA_CODE
Description	Description associated with the action code. This field is queryable. Column GKVSVBA_DESC
User ID	ID of the user who last updated the record. This field is display only. Column GKRPRCT_USER_ID
Activity Date	Date on which the record was last updated. This field is display only. Column GKRPRCT_ACTIVITY_DATE

Process Rule Set Inquiry (GKIPRST) form

Use this form to display and query the existing rule sets from the GKRPRST table. The form is not accessible through the navigation menu; it can be accessed through Direct Access or the Business Process Rule Set Form (GKRPRST).

Field	Description
Process Code	Process code. This field is queryable. Column GKRPRST_SQPR_CODE
Description	Description associated with the process code. This field is queryable. Column GKVSQPR_DESC
Rule Set Code	Rule set code. This field is queryable. Column GKRPRST_PRST_CODE
Description	Description associated with the rule set code. This field is queryable.

Field	Description
	Column GKVPRST_DESC
Rule Code	Rule code. This field is queryable. Column GKRPRST_SQRU_CODE
Description	Description associated with the rule code. This field is queryable. Column GKVSQRU_DESC
User ID	ID of the user who last updated the record. This field is display only. Column GKRPRST_USER_ID
Activity Date	Date on which the record was last updated. This field is display only. Column GKRPRST_ACTIVITY_DATE

Process Transaction Inquiry (GKIPRTR) form

Use this form to display and query the existing transaction results from the GKRPWRK table. The form is not accessible through the navigation menu; it can be accessed through Direct Access or the Process Transaction Maintenance Form (GKAPMLT).

Field	Description
Process Code	Process code. This field is queryable. Column GKRPWRK_SQPR_CODE
Description	Description associated with the process code. This field is queryable. Column GKVSQPR_DESC
Rule Set Code	Rule set code. This field is queryable. Column GKRPWRK_PRST_CODE
Description	Description associated with the rule set code. This field is queryable. Column GKVPRST_DESC
Rule Code	Rule code. This field is queryable. Column GKRPWRK_SQRU_CODE

Field	Description
Description	Description associated with the rule code. This field is queryable. Column GKVSQRU_DESC
Action Code	Action code. This field is queryable. Column GKRPWRK_SVBA_CODE
Description	Description associated with the action code. This field is queryable. Column GKVSVBA_DESC

Business Rule Builder Inquiry (GKIRSQL) form

Use this form to display and query the existing business rules from the GKRRSQL table. The form is not accessible through the navigation menu; it can be accessed through Direct Access or the Business Rules Builder Form (GKRRSQL).

Field	Description
Process Code	Process code. Column GKRRSQL_SQPR_CODE
Description	Description associated with the process code. This field is queryable. Column GKVSQPR_DESC
Rule Code	Rule code. Column GKRRSQL_SQRU_CODE
Description	Description associated with the rule code. This field is queryable. Column GKVSQRU_DESC
Active	Check box used to indicate whether the record is active. Column GKRRSQL_ACTIVE_IND
Validated	Check box used to indicate whether the SQL is valid. Column GKRRSQL_VALIDATED_IND

Field	Description
User ID	ID of the user who last updated the record. This field is display only. Column GKRRSQL_USER_ID
Activity Date	Date on which the record was last updated. This field is display only. Column GKRRSQL_ACTIVITY_DATE
SQL Statement	SQL statement associated with the selected record. This field is display only.

Business Rules Auto-Populate Rules (GKRPRCT) form

Use this form to build the rules that define the columns that will be used for process rules processing. This form defines how the actions should process each row.

Also, if the data is being written to the Process Working Table (GKRPWRK), this form allows the behavior of the Process Transaction Maintenance Form (GKAPMLT) to be defined.

In some cases, users might want the rule set results to be inserted into the GKRPWRK table. In other cases, users might want to insert the rule set results in to a different Banner table

The Auto-Population tab is used to define the table columns into which the rule set results will placed after the Business Rule Process (GKPPSQL) is run.

If a user specified that the GKRPWRK table was to be populated, then information can be entered in the Display tab. The Display tab is used to define how the rule set results from the GKRPWRK table will be displayed in the Process Transaction block on GKAPMLT.

A process/rule set is executed from the Process Launch Form (GKAPPLN) using the set of SQL statements for processing within Banner applications. The GKRPRCT form allows the user to specify how the results from the SQL statements are to be used by the actions associated with the rule set. For example, how data will be inserted into a column in the GKRPWRK table by the Auto-Populate action.

The columns defined on this form are used in business rule processing on GKAPMLT.

Key block

Users can select only records that they are allowed to use based on the information entered on the Process Rules Roles Form (GKAPRRO).

Field	Description
Process Code	Process code with which the auto-population and display information will be associated.

Field	Description
	Column GKRPRCT_SQPR_CODE
Rule Code	Rule code with which the auto-population and display information will be associated. Column GKRPRCT_SQRU_CODE
Action Code	Action code with which the auto-population and display information will be associated. Column GKRPRCT_SVBA_CODE
Summary	Select this button to display the Process Rule Column Definition Inquiry Form (GKIPRCT).

Process Rules block

The Process Rules block contains the Auto-Population tab and the Display tab.

Note: The AUTO_POPULATE action makes no assumptions and has no meta-knowledge about the tables being updated other than that provided in the GKRPRCT form. The person building the rule-set is responsible for ensuring that all Oracle constraints, Banner constrains and local business constraints are satisfied.

In particular the AUTO_POPULATE action does not automatically populate columns in the GKRPWRK table when inserting rows into that table.

Field	Description
Select Position	Indicates the position of the selected value/ column in the SQL rule statement. Column GKRPRCT_SELECT_POSITION

Auto-Population tab

The PL/SQL process action (GKKPSQL.AUTOPOPULATE) will insert a new record or update an existing record. See “AUTO_POPULATE” for details.

The fields on the Auto-Population tab provide a mapping, between the columns returned by the select clause of a SQL rule, and the columns designated as “auto-populate” using the delivered AUTO_POPULATE process action. In addition, these rules are used to provide a logical association between rule data and action data for other delivered actions and for client-defined actions.

Note: The AUTO_POPULATE action (GKKPSQL.AUTOPOPULATE) enforces strict type-validation, it does not carry out implicit type conversion. If the data type of the column referenced by the **Select Position** field (GKRPRCT_SELECT_POSITION) does not agree with the data type in the **Auto-Populate Code** field (GKRPRCT_SVAP_CODE), then a run-time exception error such as ORA-01722: Invalid Number will be raised.

The Auto-Population process action will use all of the columns, where the **Key** is set to Y, to construct a WHERE clause along with the data from the current row of the rule being processed. This WHERE clause is used to determine if a row already exists in the target table. If the row does exist, an UPDATE statement such as the following is created and executed:

UPDATE target_tableWHERE column1 = data1,column2 = data2, etc.SET columnA = dataA, columnB = dataB, etc.

Note: column1, column2 are all of the columns where **Key** = Y; columnA, columnB are all of the columns where **Key** = N)

If the row does not exist, then an INSERT statement is generated and executed to insert all the columns defined, regardless of the value of **Key** field.

Field	Description
Table	Entity (table) name to be used in the dynamically constructed SQL statements. The table must be the same for all column definition rules for a specific process/rule/action code combination. Column GKRPRCT_ENTITY
Auto-Populate Code	Table column that will be populated. The values that are entered in this field must include the table Key fields and any other mandatory fields. Column GKRPRCT_SVAP_CODE
Auto-Populate Description	Description of the auto-populate column code. The value is from GKVSVAP_DESC and can be updated. If the description is not updated, the GKRPRCT_SVAP_DESC will be left blank. The value entered here will be the auto-hint for the fields displayed in the Process Transactions block on the GKAPMLT form. Column GKRPRCT_SVAP_DESC
Key	Check box used to specify whether data in the associated column will be used when deciding whether to insert a new record or update an existing one when the rule set is executed.

Field	Description
	Y The Auto-Population process action will include this column along with the data from the current row of the rule being processed as part of a SQL WHERE clause to determine if a row already exists in the target table (from GKRPRCT_SVAP_ENTITY). N The Auto-Population process action will not include this column as part of a WHERE clause that checks if data exists. Column GKRPRCT_MERGE_IND

Display tab

For data to be displayed on the Process Transaction block on the GKAPMLT form, values must be entered in the **Abbreviation**, **Display Sequence** and **Order Sequence** fields.

Note: The rule set execution environment uses the GKRPRCT form along with the process, rule and action to find out how to handle the current row.

The GKAPMLT form uses the process code, rule code and action code to find the column definitions to know what to display and how. The GKAPMLT form also uses the process code, rule code and action code as a filter when retrieving rows from the GKRPWRK table.

Field	Description
Abbreviation	Abbreviated description to use in GKAPMLT. The text entered here will be the heading for each column of data on the Process Transaction block on the GKAPMLT form. Column GKRPRCT_SVAP_ABBR
Display Sequence	Sequence in which the column will be displayed on GKAPMLT. Column GKRPRCT_DISPLAY_SEQNO
Order Sequence	Order number in which to display the working results on GKAPLMT. The lower number indicates the highest sort priority. Column GKRPRCT_ORDER_SEQNO
Display Length	Column length that will be displayed on GKAPMLT. Column GKRPRCT_DISPLAY_LENGTH

Field	Description
Update Status	Indicates if the status can be updated on GKAPMLT. Values are as follows. `Update` Allow the status to be updated `No Update` Do not allow the status to be updated `Update if Null` Status is only updated if it was null. Column `GKRPRCT_UPDATE_STATUS`
Order By	Indicates if the rows should be sorted by ascending (default) or descending values on GKAPMLT. Column `GKRPRCT_ORDER_ASCDESC`
Format	Justification/precision for the **Number** column, or the date format for the **Data** column displayed on the GKAPMLT form. The **Format** field must have a number sign (#), which acts as a placeholder for the column value. If the column data type is `NUMBER`, then the field value must be `TO_CHAR(#, 's9999')` or `TO_NUMBER(TO_CHAR(#,'999999999'))` where `s9999` and `'999999999'` are any valid number format. If the column data type is `DATE`, then the field value must be `TO_CHAR(#,'YYYYMMDD')` where `'YYYYMMDD'` is any valid date format. A value can only be entered in the **Format** field if the column data type is either `NUMBER` or `DATE`. The GKAPMLT form will use a PL/SQL function to apply a format or `TO_CHAR` conversion to the field. The # character in the field will be replaced by the value of the appropriate column. This is shown in the following example: `TO_CHAR(#,'YYYYMMDD')` will display a date as 20070323. `TO_CHAR(#, 's9999')` will display an integer with a + or - prefix. `TO_NUMBER(TO_CHAR(#,'999999999'))` will display a number as right justified.

Field	Description
	Column GKRPRCT_SELECT_POSITION
Entity	Table used for selecting and validating data entered in the GKAPMLT form. When a table name is entered here, the column on the Process Transaction block on the GKAPMLT form will have a List of Values (LOV) that corresponds to that table. This field supports the use of any Banner validation table and the SMRPRLE table.
Attribute	Value used for selected and validation data entered in the GKAPMLT. This is used with the **Entity** value. If a value is entered for the **Entity** and **Attribute** fields, then the GKAPMLT form will create a LOV query against the GTVSDAX table as follows: • The description that is displayed is from the GTVSDAX_DESC column. • The code to be inserted is from the GTVSDAX_EXTERNAL_CODE column. The rows that are selected are where the following are true: GTVSDAX_INTERNAL_CODE = entity and GTVSDAX_INTERNAL_CODE_GROUP = attribute

Business Process Rule Set (GKRPRST) form

Use this form to assign sub-tasks (rule codes) and actions to a task (rule set), and assign the desired parameter values.

Multiple rules can be added to the process code/rule set combination, which will allow the SQL statements (from GKRRSQL) to be executed in sequence (see Rule set execution modes on page 24 for details on how the rule set can be executed). Actions can also be added to the process code/rule set combination, which will associate a PL/SQL procedure with the rule set.

Rules are used to support SELECT statements (such as “select data using the criteria entered in the SQL statement”). Actions are used to support UPDATE and DELETE statements (such as “update the data in a table”), or used to support other procedures such as transmitting an e-mail message or calling a Banner API. The design of the software allows clients to develop local actions to support their own processing needs.

For example, a rule set could contain a number of rules to do the following:

- Create a group of students and (using an action) assign them to a cohort.
- Calculate an overall mark or grade for each student and use an action to store the information as a process transaction (GKRPWRK).
- Use the overall mark or grade to calculate a progression recommendation and store the information as a process transaction.

The parameters defined on the Business Rule Parameters Form (GKRSQRP) and Business Rule Action Parameters Form (GKRSVBA) will be the default parameters on GKRPRST. The default values can be overridden.

After the desired rules and actions are added to a process code/rule set combination, the GKAPPLN form can be used to execute the rule set and the attached rules and actions, or the rule can be executed by the APIs (see APIs on page 44).

See Assign rules and actions to rule sets on page 34.

Key block

The Key block contains the following fields.

Field	Description
Process Code	Process code with which the rules and actions will be associated. Users can only select process codes that they are allowed to use based on the information entered on the Process Rules Roles Form (GKAPRRO). Column `GKRPRST_SQPR_CODE`
Rule Set	Rule set code with which the rules and actions will be associated. Column `GKRPRST_PRST_CODE`
Copy From	Opens the Copy From window, where the user can enter a default process and rule set if the process and rule set in the Key block have no records.
Summary	Select this button to display the Process Rule Set Inquiry Form (GKIPRST).

Copy Parameters window

This window is used to enter a default process and rule set if the process and rule set in the Key block have no records. This window is opened by clicking the **Copy From** button.

Field	Description
Copy from Process	Process code from which to copy the parameters. The LOV displays all of the existing process/rule set combinations that exist in GKRPRST. Column `GKRPRST_SQPR_CODE`
Copy From Rule Set	Rule set code from which to copy the parameters. The LOV displays all of the existing process/rule set combinations that exist in GKRPRST. Column `GKRPRST_PRST_CODE`
Copy	Copies the records from the **Copy From Process** and **Copy From Rule Set** fields to the new process/rule set combination.
Cancel	Returns to the Key block. No records are copied.

Process Rule Set and Parameters tab

This tab contains the Rule Set Rules block and Rule Set Rule Parameters block.

Rule Set Rules block

The Mass Data Update Utility only supports only rules that contain `SELECT` statements. If you want to create `UPDATE` or `DELETE` statements, you need to create an action on the Rule Set Actions and Parameters tab. You can create as many rule sets as necessary.

Field	Description
Run Sequence	Sequence number, which indicates the order in which the rule will be run. Column `GKRPRST_SEQ_NO`
Rule	Rule code that describes what the SQL statement will do. Users scan only select rule codes that they are allowed to use based on the information entered on the GKAPRRO form.

Field	Description
	Column GKRPRST_SQRU_CODE
Rule Description	Description of the rule code. Display only. The value is from GTVSQRU_DESC.
Active	Indicates if the SQL statements for the rule are active. • Selected = (default) Rules are active • Cleared = Rules are not active Column GKRPRST_ACTIVE_IND
Number of Sequences	Indicates the number of SQL statements for the selected rule (GKRPRST_SQRU_CODE). The count will be derived from the number of sequences for a rule as defined on the Business Rules Builder Form (GKRRSQL) Display only. Column GKRPRST_RULE_NUMBER_SEQUENCES
Activity Date	Date when the record was created or last updated. Display only. Column GKRPRST_ACTIVITY_DATE
User ID	Banner ID of the person who created or last updated the record. Display only. Column GKRPRST_USER_ID

Rule Set Rule Parameters block

The parameters are associated with the rule selected in the Rule Set Rules block. For parameters to exist in this block for the selected rule, parameters must be defined on the GKRSQRP form for the same process code/rule code combination.

These parameters can be changed in the GKAPPLN form before running the GKPPSQL process.

Field	Description
Revert All	Returns all of the values in the **Default Value** field to the values from GKRSQRP.
Parameter	Parameter code associated with the selected rule. The value is from GKRSQRP_SQPA_CODE. Column GKRPRSP_SQPA_CODE
Parameter Description	Description of the parameter code. Display only. The value is from GKVSQPA_DESC.

Field	Description
Default Value	Default value for the parameter that the rule will use when processing. This value will override the value from the GKRSQRP table. Column GKRPRSP_DEFAULT_VALUE
Revert	Changes the value in the **Default Value** field for the selected parameter to the value from GKRSQRP.
Start Date	Start date for the process rule set parameter. Display only. The value is from GKVSQRU_START_DATE.
End Date	End date for the process rule set parameter. Display only. The value is from GKVSQRU_END_DATE.
Activity Date	Date when the record was created or last updated. Display only. Column GKRPRSP_ACTIVITY_DATE
User ID	Banner ID of the person who created or last updated the record. Display only. Column GKRPRSP_USER_ID

Rule Set Actions and Parameters tab

This tab contains the Rule Set Actions block and Rule Set Action Parameters block.

Rule Set Actions block

Use this block to associate actions with a rule set. You can add as many actions as needed, but the action codes must be unique within the rule sets.

Actions contain PL/SQL procedures that will carry out tasks such as updating or inserting data.

Field	Description
Sequence	Order in which the action(s) will be performed. Column GKRPACT_SEQNO
Action	Action code that determines what the set of SQL statements will do.

Field	Description
	Users can only select action codes that they are allowed to use based on the information entered on the GKAPRRO form. The value is from GKVSVBA_SVBA_CODE. Column GKRPACT_SVBA_CODE
Action Description	Description of the action code. The value is from GKVSVBA_DESCDisplay only.
Active	Indicates if the rule set action is active. • Selected = (default) Active • Cleared = Not active Column GKRPACT_ACTIVE_IND
Function	Database function and procedure name. Display only. The value is from GKVSVBA_DBPROC_NAME.
Start Date	Action start date. Display only. The value is from GKVSVBA_START_DATE.
End Date	Action end date. Display only. The value is from GKVSVBA_END_DATE.
Activity Date	Date when the record was created or last updated. Display only. Column GKRPACT_ACTIVITY_DATE
User ID	Banner ID of the person who created or last updated the record. Display only. Column GKRPACT_USER_ID

Rule Set Action Parameters block

The parameters are associated with the action selected in the Rule Set Actions block. For parameters to exist in this block for the selected action, parameters must be defined on the GKRSVBA form for the same process code/action code combination.

Note: Action parameters will be effective only for rule/actions that are specifically written to support parameter actions. In most cases, using the Rule Set Rule parameter is easier and more flexible than using the Rule Set Action parameters.

These parameters can be changed from GKAPPLN before running the GKPPSQL process.

Field	Description
Revert All	Returns all of the values in the **Default Value** field to the values from GKRSVBA.
Parameter	Parameter code associated with the selected action in the Rule Set Actions block. Display only. The value is from GKRSVBA_SQPA_CODE. Column GKRPACP_SQPA_CODE
Parameter Description	Description of the parameter code. The value is from GKVSQPA_DESC.
Default Value	Default value for the parameter that the action will use when processing. This value will override the value from the GKRSVBA table. Column GKRPACP_DEFAULT_VALUE
Revert	Changes the value in the **Default Value** field for the selected parameter to the value from GKRSVBA.
Start Date	Start date for the parameter. Display only. The value is from GKVSVBA_START_DATE.
End Date	End date for the parameter. Display only. The value is from GKVSVBA_END_DATE.
Activity Date	Date when the record was created or last updated. Display only. Column GKRPACP_ACTIVITY_DATE
User ID	Banner ID of the person who created or last updated the record. Display only. Column GKRPACP_USER_ID

Business Rules Builder (GKRRSQL) form

Use this form to build dynamic SQL statements that will be used for processing the sub tasks (rule codes). The SQL statements are assigned to a process code/rule code combination. When the

process code/rule code combination is executed from the GKAPPLN form, the code in the SQL statement will be read and executed.

The form allows you to specify the columns you want to use in your statements, and provides operators so you can build them. You must validate, activate, and save each SQL statement before you can use it.

Oracle has a limitation of 30 characters on the length of column names when processing dynamic SQL. When a SQL function or SQL operators are being used to derive a column value from other columns, and the length of this is greater than 30 characters, you will need to assign an alias to the column. For example, a sql rule might contain:

```
SELECT 'University of Banner'||nvl(sgbstdn_program_1,'no program
 name'), sgbstdn_pidm ...
```

Oracle will assume the column name for the first column is the same as the column definition, which is more than 30 characters and will result in the dynamic SQL process failing. To overcome this issue, assign an alias to the column as follows:

```
SELECT 'University of Banner'||nvl(sgbstdn_program_1,'no program name')
 programme, sgbstdn_pidm ...
```

After the SQL statement is created, the statement must be validated to be correctly processed in the Process Launch Form (GKAPPLN). You can save GKRRSQL records without validating them and return to them later, but you cannot use them until you validate and activate them. If you modify a rule that was previously validated, the rule becomes invalid until you re-validate it by clicking **Validate** to invoke the SQL parsing process again. When the record is saved, the **User ID** field is updated. If a user, other than the user who last validated the SQL, saves the record, the **Validated** check box is cleared. The SQL will then need to be re-validated and will pick up the validating user ID.

Clients may prefer to develop and test the SQL rules using one of the many SQL development tools; and then copy and paste the SQL into GKRRSQL.

The following must be true for a rule to be used:

- The rule must be validated
- The rule must be activated
- The current date must be within the start and end dates for the rule

Note: A business rule consists of one or more SQL statements that are executed sequentially when a rule set containing the rule is executed.

Key block

The Key block contains the following fields.

Field	Description
Process Code	Process code with which the SQL statement will be associated. Users can only select process codes that they are allowed to use based on the information entered on the Process Rules Roles Form (GKAPRRO). Column `GKRRSQL_SQPR_CODE`
Rule Code	Rule code that describes what the SQL statement will do. Users can only select rule codes that they are allowed to use based on the information entered on the GKAPRRO form. Column `GKRRSQL_SQRU_CODE`
Summary	Select this button to display the Business Rule Builder Inquiry Form (GKIRSQL).

Rule Data block

To be able to use rule data, ensure that the date is between the values in the **Start Date** and **End Date** fields, and the **Active** check box is selected.

Field	Description
Sequence	Sequence number for the rule. The sequence determines the order in which the statements are processed. A business rule can have multiple sequences. Display only. Column `GKRRSQL_SEQ_NO`
User ID	Banner ID of the person who created or last updated the record. Display only. Column `GKRRSQL_USER_ID`
Start Date	Date the rule becomes active. The start date is required and the default value is the system date. Column `GKRRSQL_START_DATE`

Field	Description
End Date	Date the rule becomes inactive. The end date can be null to signify the end of time. Column GKRRSQL_END_DATE
Activity Date	Date when the record was created or last updated. Display only. Column GKRRSQL_ACTIVITY_DATE
SQL Statement	SQL statement for this process code/rule code sequence. You can use the **Table**, **Column**, **Operator** and **Parameter** buttons and fields to help build the statement. Each selection from the buttons and fields is appended to the SQL statement. You can enter any parameter/bind variable, however, the SQL will be validated only if attached to the same process code on GKRSQPA. It is recommended that you use a SQL development tool to create complex SQL statements. Column GKRRSQL_WHERE_CLAUSE The AUTO_POPULATE action makes no assumptions and has no meta-knowledge about the tables being updated other than that provided in the GKRPRCT form. The person building the rule-set is responsible for ensuring that all Oracle constraints, Banner constrains and local business constraints are satisfied.In particular, the AUTO_POPULATE action does not automatically populate columns in the GKRPWRK table when inserting rows into that table.
Validate	Validates the SQL statement. The SQL statement must be validated before it can be processed on the Process Launch Form (GKAPPLN). The record must be saved before the SQL can be validated.
Active	Indicates if the rule is currently active. • Selected = Rule is active • Cleared = Rule is not active Column GKRRSQL_ACTIVE_IND

Field	Description
Validated	Indicates that the SQL statement has been successfully validated, and will be correctly processed on GKAPPLN. You cannot change the **Validated** indicator; Banner updates it automatically when the SQL statement is validated. Column `GKRRSQL_VALIDATED_IND`
Table	Tool to help build the SQL statement. Click the button to see a list of Banner tables, then double-click on the field to append the table name to the SQL statement.
Column	Tool to help build the SQL statement. Click the button to see a list of columns for the selected table, then select the column you want. The column will be appended to the SQL statement.
Operator (untitled)	Tool to help build the SQL statement. Select the operator you want to use in your SQL statement from the pull-down list. The operator will be appended to the SQL statement.
Parameter	Tool to help build the SQL statement. Click the button to see a list of parameters that were defined for the selected process code on the GKRSQPA form, then select the parameter you want. The parameter will be appended to the SQL statement.
Editor	Displays the Banner Editor so that you can make additional changes to your SQL statement.

Parsed SQL block

This block is displayed when you select Next Block from the Rule Data block. It contains the current, parsed version of your SQL. This view may be helpful during updates, when you try to compare changes to existing parsed, error-free SQL.

Business Rules Process Parameters (GKRSQPA) form

Use this form to associate the process code with the parameter codes that will be used on Business Rules Builder Form (GKRRSQL).

The parameters entered on this form for a process code can be selected on the GKRSSQL form when the **Parameter** LOV button is clicked. The only parameters that can be entered on the

GKRSSQL form for a process/rule code combination are the parameters attached to the same process code on the GKRSQPA form.

Key block

The Key block contains the following fields.

Field	Description
Process Code	Process code to which you want to assign parameters. Column GKRSQPA_SQPR_CODE

Rules block

This block allows you to specify the parameters for the process code.

Field	Description
Parameter Code	Parameter code you want to associate with the process code. Multiple parameters can be assigned to a process code. Column GKRSQPA_SQPA_CODE
Description	Description of the parameter code. Display only.
System Required Indicator	Indicates if the code is required by the system. • Selected = The code is required for Banner to process the data correctly. A system required code cannot be deleted. • Cleared = The code is not required by the system. Column GKRSQPA_SYS_REQ_IND
User ID	Banner ID of the person who created or last updated the record. Display only. Column GKRSQPA_USER_ID
Activity Date	Date when the record was created or last updated. Display only. Column GKRSQPA_ACTIVITY_DATE

Business Rule Parameters (GKRSQRP) form

Use this form to associate the process code and rule code with the parameter codes that will be used on Business Rules Builder Form (GKRRSQL).

The parameters entered on the GKRSQRP form will be the default parameters in the Rule Set Rule Parameters block on the GKRPRST form, when the process code/rule code combination is the same for both forms. The default value may be overwritten in GKRPRST.

Key block

The Key block contains the following fields.

Field	Description
Process Code	Process code to which you want to assign parameters. Column `GKRSQRP_SQPR_CODE`
Rule Code	Rule code to which you want to assign parameters. Column `GKRSQRP_SQRU_CODE`

Base block

This block allows you to specify the parameters for the process code and rule code.

Field	Description
Parameter Code	Parameter code you want to associate with the process and rule codes. Multiple parameters can be assigned. Column `GKRSQRP_SQPA_CODE`
Description	Description of the parameter code. Display only.
System Required Indicator	Indicates if the code is required by the system. • Selected = The code is required for Banner to process the data correctly. A system required code cannot be deleted. • Cleared = The code is not required by the system. Column `GKRSQRP_SYS_REQ_IND`

Field	Description
Default Value	Default value for the parameter that the rule code will use when processing. Column GKRSQRP_DEFAULT_VALUE
User ID	Banner ID of the person who created or last updated the record. Display only. Column GKRSQRP_USER_ID
Activity Date	Date when the record was created or last updated. Display only. Column GKRSQRP_ACTIVITY_DATE

Business Rule Action Parameters (GKRSVBA) form

Use this form to define the action parameter codes to be used with a business action function. The business action parameters and default values will be defined, and then executed by the business action function.

The parameters entered on the GKRSVBA form will be the default parameters in the Rule Set Action Parameters block on the GKRPRST form, when the process code/action code combination is the same for both forms.

Key block

The Key block contains the following fields.

Field	Description
Process Code	Process code to which you want to assign parameters. Column GKRSVBA_SQPR_CODE
Action Code	Action code to which you want to assign parameters. Column GKRSVBA_SVBA_CODE
Function	Function or procedure.package associated with the action code. Display only. The value is from GKVSVBA_DBPROC_NAME.

Base block

This block allows you to specify the parameters for the process code and action code.

Field	Description
Parameter Code	Parameter code you want to associate with the action function. Multiple parameters can be assigned. Column GKRSVBA_SQPA_CODE
Description	Description of the parameter code. Display only.
System Required Indicator	Indicates if the code is required by the system. • Selected = The code is required for Banner to process the data correctly. A system required code cannot be deleted. • Cleared = The code is not required by the system. Column GKRSVBA_SYS_REQ_IND
Default Value	Default value for the parameter that the action function will use when processing. Column GKRSVBA_DEFAULT_VALUE
User ID	Banner ID of the person who created or last updated the record. Display only. Column GKRSVBA_USER_ID
Activity Date	Date when the record was created or last updated. Display only. Column GKRSVBA_ACTIVITY_DATE

Process Activity Code Validation (GKVPACT) form

Use this form to enter or display codes for activities assigned to processing tasks.

Note: An activity that is associated with the source BASELINE cannot be modified. It can be duplicated into a local copy, which you can then modify.

Field	Description
Activity Code	Code of the activity.

Field	Description
	Column GKVPACT_ACTI_CODE
Description	Description of the activity. Column GKVPACT_DESC
Process Code	Code of the process with which the activity is associated. Column GKVPACT_SQPR_CODE
Source	Source of the record. Valid values are the following. Baseline The record was a delivered value. Local The record was created locally. Temporary The record was created during an import/export activity and will be purged automatically. Column GKVPACT_SOURCE_IND
User ID	ID of the user who last updated the record. This field is display only. Column GKVPACT_USER_ID
Activity Date	Date on which the record was last updated. This field is display only. Column GKVPACT_ACTIVITY_DATE

Business Rule Set Code Validation (GKVPRST) form

Use this form to identify the rule set codes available for processing. The rule set codes are used to define a task within a process (that is, rule sets are the executable objects).

After a code is used in an existing table, the record cannot be deleted.

Field	Description
Code	Rule set code. Column GKVPRST_CODE
Description	Description of the rule set code. Column GKVPRST_DESC

Field	Description
Execute Mode	Indicates how a rule set with multiple rules and multiple actions will be executed from the Business Rule Process (GKPPSQL). Valid values are as follows. `RULE` Rule set is to be processed in RULE mode `ACTION` Rule set is to be processed in ACTION mode Refer to Rule set execution modes on page 24 for detailed information about these modes. Column `GKVPRST_EXECUTE_BY`
Start Date	Date the code becomes active. The start date is required and the system date is the default. Column `GKVPRST_START_DATE`
End Date	Date the code becomes inactive. The end date can be null to signify the end of time. Column `GKVPRST_END_DATE`
Activity Date	Date when the record was created or last updated. Display only. Column `GKVPRST_ACTIVITY_DATE`
User ID	Banner ID of the person who created or last updated the record. Display only. Column `GKVPRST_USER_ID`

Business Rule Parameter Code Validation (GKVSQPA) form

Use this form to define the parameters or variables used to build the process rules on the Business Rules Builder Form (GKRRSQL). The codes defined on this form will be used in the dynamic SQL statements on GKRRSQL.

All parameter codes used in GKRRSQL must be defined on GKVSQPA, including parameter codes used by a rule, which are defined in the Business Rule Process Code Validation Form (GKVSQPR).

The `TERM` and `PIDM` codes are delivered as seed data but are not required for system processing.

Field	Description
Code	Process parameter code. Column `GKVSQPA_CODE`

Field	Description
Description	Description of the parameter code. Column GKVSQPA_DESC
Data Type	Data type associated with the parameter code. Valid values are Character, Number and Date. Column GKVSQPA_DATA_TYPE_CDE
Start Date	Date the code becomes active. The start date is required and the system date is the default. Column GKVSQPA_START_DATE
End Date	Date the code becomes inactive. The end date can be null to signify the end of time. Column GKVSQPA_END_DATE
Activity Date	Date when the record was created or last updated. Display only. Column GKVSQPA_ACTIVITY_DATE
User ID	Banner ID of the person who created or last updated the record. Display only. Column GKVSQPA_USER_ID

Business Rule Process Code Validation (GKVSQPR) form

Use this form to define the process codes available for dynamic query processing. Process codes are used to describe high-level processes. The process code is the code to which the rules and rule sets will be attached.

Examples

- UPR - Undergraduate Progression Recommendation
- BASE_PROGRESSION - Progression

The BASE_PROGRESSION code is delivered as seed data. Only the description of the BASE_PROGRESSION code can be updated.

Codes that have the **System Required** check box selected, or that are used in an existing table cannot be deleted.

Field	Description
Code	Process code for dynamic SQL processing. The code represents the function of the process. Column GKVSQPR_CODE
Description	Description of the process code. Column GKVSQPR_DESC
System Required	Indicates if the code is required by the system. • Selected = The code is required for Banner to process the data correctly. A system required code cannot be deleted. • Cleared = The code is not required by the system. Column GKVSQPR_SYS_REQ_IND
Start Date	Date the process code becomes active. The start date is required and the system date is the default. The date is for information purposes only. The date will not restrict when the process code can be used. Column GKVSQPR_START_DATE
End Date	Date the process code becomes inactive. The end date can be null to signify the end of time.The date is for information purposes only. The date will not restrict when the process code can be used. Column GKVSQPR_END_DATE
Activity Date	Date when the record was created or last updated. Display only. Column GKVSQPR_ACTIVITY_DATE
User ID	Banner ID of the person who created or last updated the record. Display only. Column GKVSQPR_USER_ID

Business Rule Code Validation (GKVSQRU) form

Use this form to define the rule codes that will be used in dynamic SQL statements. The rule codes describe common or unique rules that will perform a sub-task.

The rule codes contain the SQL that is executed to retrieve the data when executing a rule set.

Examples

- Min Aggregate Weighted Pass Mark >60%
- Min Pass Mark SubjB (mathematics B) >60%
- Min Pass Mark Course Work >45%

Records cannot be deleted if the code is used on the Business Rules Builder Form (GKRRSQL).

Field	Description
Code	Rule code that identifies the rule(s) used on GKRRSQL. Column GKVSQRU_CODE
Description	Description of the rule code. Column GKVSQRU_DESC
System Required	Indicates if the code is required by the system. • Selected = The code is required for Banner to process the data correctly. A system required code cannot be deleted. • Cleared = The code is not required by the system. Column GKVSQRU_SYS_REQ_IND
Start Date	Date the rule code becomes active. The start date is required and the system date is the default. Column GKVSQRU_START_DATE
End Date	Date the rule code becomes inactive. The end date can be null to signify the end of time. Column GKVSQRU_END_DATE
Activity Date	Date when the record was created or last updated. Display only. Column GKVSQRU_ACTIVITY_DATE
User ID	Banner ID of the person who created or last updated the record. Display only. Column GKVSQRU_USER_ID

Auto-Populate Code Validation (GKVSVAP) form

Use this form to define the fields in the GKRPWRK table that will be populated automatically (auto-populated) from other Banner tables. The data entered here will determine what can be selected in the **Auto-Populate Code** field on the Business Rules Auto-Populate Rules Form (GKRPRCT).

Seed data is delivered for this form. Only the descriptions of the delivered codes can be changed.

Codes that have the **System Required** check box selected, or that are used in an existing table cannot be deleted.

Field	Description
Code	Database column that will be automatically populated (for example, the `GKRPWRK_V01` column may be used to store a student's programme code). Column `GKVSVAP_CODE GKVSVAP_DESC`
System Required	Indicates if the code is required by the system. • Selected = The code is required for Banner to process the data correctly. A system required code cannot be deleted. • Cleared = The code is not required by the system. Column `GKVSVAP_SYS_REQ_IND`
Start Date	Date the code becomes active. The start date is required and the system date is the default. Column `GKVSVAP_START_DATE`
End Date	Date the code becomes inactive. The end date can be null to signify the end of time. Column `GKVSVAP_END_DATE`
User ID	Banner ID of the person who created or last updated the record. Display only. Column `GKVSVAP_USER_ID`
Activity Date	Date when the record was created or last updated. Display only. Column `GKVSVAP_ACTIVITY_DATE`

Business Action Code Validation (GKVSVBA) form

Use this form to define the action codes and the procedures they will execute. The action codes will launch the task (rule set) and sub-tasks (rules) within a process.

The action codes and associated procedures/functions can be used with a rule set code to process progression groupings, recommendations and student record updates.

Examples

- AUTO_POPULATE carries out generic update tasks.
- CALC_RECOMMENDATION starts the process that gathers and executes the required recommendation rules to produce a dynamically defined output.

The user can add codes and create locally defined generic or specific PL/SQL procedures. Codes that have the **System Required** check box selected or that are used in an existing table cannot be deleted.

Seed data is delivered for this form. Only the descriptions of the delivered codes can be changed.

Field	Description
Code	Business action code and description that determines what the set of SQL statements will do. Column `GKVSVBA_CODE GKVSVBA_DESC`
System Required	Indicates if the code is required by the system. • Selected = The code is required for Banner to process the data correctly. A system required code cannot be deleted. • Cleared = The code is not required by the system. Column `GKVSVBA_SYS_REQ_IND`
Package/Procedure	Package.procedure that will perform certain actions when called by the rule. Column `GKVSVBA_DBPROC_NAME`
Start Date	Date the code becomes active. The start date is required and the system date is the default. Column `GKVSVBA_START_DATE`
End Date	Date the code becomes inactive. The end date can be null to signify the end of time. Column `GKVSVBA_END_DATE`
User ID	Banner ID of the person who created or last updated the record. Display only.

Field	Description
	Column GKVSVBA_USER_ID
Activity Date	Date when the record was created or last updated. Display only. Column GKVSVBA_ACTIVITY_DATE

Tables

The following tables are part of the Mass Data Update Utility release.

- Job Submission Base Output Table (GKBOUTP)
- Working Run Table (GKBPSQL)
- Job Submission Repeating Output Table (GKROUTP)
- Process Rule Set Action Parameters Table (GKRPACP)
- Process Rule Set Actions Table (GKRPACT)
- Activity Join/Filter Table (GKRPAJF)
- Activity Sources Table (GKRPASO)
- Activity Target Column Definitions Table (GKRPATC)
- Activity Target Column Rule Table (GKRPATS)
- Activity Unique Target Table (GKRPATT)
- Process Execution Trees Table (GKRPPTR)
- Process Code Business Action Rules Table (GKRPRBA)
- Process Rule Column Definition Table (GKRPRCT)
- Process Code Rules Table (GKRPRRO)
- Process Code Business Rule Rules Table (GKRPRRU)
- Business Process Rule Set Parameter Table (GKRPRSP)
- Business Process Rule Set Table (GKRPRST)
- Process Parameters Table (GKRPSQL)
- Process Working Table (GKRPWRK)
- Diagnostic Table (GKRRLOG)
- Business Rule Builder Table (GKRRSQL)
- Business Rule Process Parameters Table (GKRSQPA)
- Business Rule SQL Parameters Table (GKRSQRP)
- Business Action Parameters Table (GKRSVBA)
- SEVIS SQL Rules Table (GKRSVSQ)
- Last User Parameter Values Table (GKRUPRM)
- Working Item LOV Setup Temporary Table (GKTPLOV)
- Activity Codes Table (GKVPACT)
- Boolean Values Table (GKVPBOO)
- Column Data Types Table (GKVPDTY)
- Business Process Rule Set Code Validation Table (GKVPRST)
- Source Indicators Table (GKVPSOU)

- Activity Source Synonyms Table (GKVPSYN)
- Task Codes Table (GKVPTAS)
- Business Rule Parameter Code Table (GKVSQPA)
- Business Rule Process Code Validation Table (GKVSQPR)
- SQL Rule Code Validation Table (GKVSQRU)
- Auto-Populate Code Validation Table (GKVSVAP)
- Business Action Code Validation Table (GKVSVBA)

Web Tailor Seed Data

The following rows are inserted into TWGBWMNU table for Web Tailor. These seed data values are required to show the exported tables or activities in XML or CSV files through browsers.

Field	Description
gkkoutp.csv	Mass Data Update Utility CSV output - for generating CSV output of the data
gkkpxml.P_Showxml	Mass Data Update Utility XML output - for generating XML output of the data.

Glossary

These terms are used throughout MCUU.

Action

Action that will be taken for each row of data produced by the SQL rules when a rule set is executed. Actions are used to execute rules to carry out particular tasks (rule sets).

Auto-Populate Code

Defines the fields that will be populated automatically (auto-populated) when a specific action is launched.

Parameter Code

Run-time parameters can be associated with rules and rule sets. When the rule set is executed, the parameters are requested from the end user and ‘bound’ into the rules. For example, a rule set for academic progression might have parameters for term code, programme and major, which would allow the rule set to be run for a particular term, programme or major.

Process

Process codes describe a high-level process. Rules and rule sets will be assigned to a process.

Rule Code

For example, distributing passing grades, and distributing the highest grades.

Rule Set

A rule set contains the codes that will describe a task within a process.

SQL Statement

SQL statements are used to process the sub-tasks (rule codes).