Banner General

Technical Reference Manual

March 2020

Notices and Privacy

© 1993-2020 Ellucian.

Contains confidential and proprietary information of Ellucian and its subsidiaries. Use of these materials is limited to Ellucian licensees, and is subject to the terms and conditions of one or more written license agreements between Ellucian and the licensee in question.

In preparing and providing this publication, Ellucian is not rendering legal, accounting, or other similar professional services. Ellucian makes no claims that an institution's use of this publication or the software for which it is provided will guarantee compliance with applicable federal or state laws, rules, or regulations. Each organization should seek legal, accounting, and other similar professional services from competent providers of the organization's own choosing.

Ellucian's Privacy Statement is available at: www.ellucian.com/privacy.

Ellucian shall have the right to (a) use, store, process, modify, reproduce, distribute and display customer data, and to grant sublicenses to third parties, for the sole purposes of providing the software, performing Ellucian's obligations under its agreements with customers and complying with applicable law or legal requirements; (b) use, store, process, modify and reproduce customer data for Ellucian's internal business purposes, including development, diagnostic, forecasting, planning, analysis and corrective purposes in connection with the software, and for otherwise improving and enhancing the software; and (c) use, store, process, modify, reproduce, display, perform, distribute, disclose and otherwise exploit in any manner Aggregated Data for Ellucian's business purposes, including disclosure within its public statements and marketing materials describing or promoting Ellucian or the software. “Aggregated Data” means any data obtained or generated by Ellucian, including data pertaining to the software, Ellucian's systems and software, and the use of any of the foregoing, and includes data derived from customer data, which in all instances (i) does not identify any individual and (ii) is not attributed or attributable to a specific customer. Aggregated Data includes data that has been combined into databases which include third party data.

Ellucian
2003 Edmund Halley Drive
Reston, VA 20191
United States of America

Contents

Banner Standards

This chapter discusses the naming standards in Banner.

Naming Banner objects

Banner page, report, job, and table names have a 7-character structure. The first and second characters identify the system and module, the third character identifies the type of object, and the four remaining characters are used as a unique identifier for the object.

These naming standards, and the meanings of each letter in the first, second, and third positions, are detailed in Chapter 1, "Overview," of the *Banner Getting Started Guide*.

For API naming standards, see Chapter 1, "Overview", of the *Banner API Developer Guide*.

Naming of Client-Developed Items

The letters W, Y, and Z are reserved for use in Positions 1 and 2 of the names of all client-developed applications, forms, reports, tables and modules.

For client-developed new applications built to coexist with Banner applications, W, Y, or Z should be used as the first character.

For client-developed forms or modules used within a Banner application, the system identifier is used as the first character (for example, G for General), and W, Y, or Z should be used as the second character.

Note: After you create a custom form, be sure to access the Object Maintenance Form (GUAOBJS) and associate it with a System indicator code, (for example, A for Advancement, G for General, F for Finance, etc.) These codes are defined on the System Indicator Validation Form (GTVSYSI). If you want to classify your form as a custom form rather than associating it with a Banner system, you can set up W, Y, and Z on GTVSYSI and use that code on GUAOBJS. If you set up a code other than W, Y or Z on GTVSYSI and use it on GUAOBJS, it is possible that Banner may not display your custom form on the appropriate menus.

For client-developed reports, tables and programs used within a Banner application, the system identifier must be used as the first character (for example, G for General and so forth), and W, Y, or Z should be used as the second character.

Column names

Column names start with the seven-character table name, followed by an underscore and an expression that uniquely identifies the column within the table.

For example:

`GJBJOBS_NAME`

`APBCONS_PIDM`

Application tables (base/repeating)

Column names that correspond to a validation table must contain the seven-character application table name followed by an underscore, the four-character validation table identifier, an underscore, and `CODE`

For example:

`GJBJOBS_PRNT_CODE`

`APRCATG_DONR_CODE`

If multiple columns are needed for the same validation table identifier, column names are made unique by appending a number or a unique name to the end of the name of the column. For example:

`GURFEED_PAYT_CODE`

`GURFEED_PAYT_CODE_TRANSCRIPT`

`APBCONS_ATYP_CODE_PREF`

`APBCONS_ATYP_CODE_CM`

The name of the last activity date column begins with the seven-character table name followed by an underscore and `ACTIVITY_DATE`. For example:

`GTVLETR_ACTIVITY_DATE`

`APBCONS_ACTIVITY_DATE`

The name of the updating user ID column begins with the seven-character table name followed by an underscore and `USER_ID`. For example:

`GURAPAY_USER_ID`

Validation tables

The validation table and corresponding page have the same name. Both start with `GTV` followed by a unique four-character identifier.

For example:

GTVCALL

The name of the key column begins with the seven-character table name followed by an underscore and `CODE`. For example:

`GTVCALL_CODE`

The name of the description column begins with the seven-character table name followed by an underscore and `DESC`. For example:

`GTVCALL_DESC`

The name of columns that are used as indicators begins with the seven-character table name and end with an underscore and `IND`. For example:

`GTVCALL_DUPL_IND`

The name of the last activity date column begins with the seven-character table name followed by an underscore and `ACTIVITY_DATE`. For example:

`GTVCALL_ACTIVITY_DATE`

A unique index is created for the validation table using the key columns to prevent duplicates from being added to the system.

Database programming object naming standards

This section discusses the naming standards for database programming objects.

The dbprocs directory

The `dbprocs` directory, found under each product directory, stores the database programming object `create` scripts (for triggers, packages, etc.). This directory also stores Banner APIs (see Chapter 7, “APIs”).

Scripts that create triggers

All scripts in the `dbprocs` directory that pertain to the creation of database triggers are named using the following standard.

`abcdddde.sql`

`a`= Product identifier (S)tudent, (P)ayroll etc.

`b`= Module (E)mployee, (B)udget etc.

`c`= (T)rigger

`dddd` = Table identifier such as PERS, IDEN, EMPL etc.

`e`= Number 0 through 9, letters a through z

Note: This becomes `aabcddde.sql` for those products that have a double character identifier. They sacrifice one of the table identifier letters: `dddd` becomes `ddd`.

The script has the same name as the table except that the third position is replaced with the letter `t` to denote a trigger. Each script ends with a number so the programming logic can have multiple triggers for the same table. If there are more than 10 triggers for a table, each script ends in a letter. For example:

`sptpers0.sql` - First database trigger for the SPBPERS table `sptiden7.sql` - Eighth database trigger for the SPRIDEN table

`petemplc.sql` - Thirteenth database trigger for the PEBEMPL table

Duplicate names

The standards for script names could potentially lead to duplicate names from time to time.

For example, if a trigger is created for both the NBBJOBS table and the NBRJOBS table you end up with two create scripts that should be named nbtjobs0.sql. This will not occur often, but when it does a small modification to one or both of the script names is suggested to make them unique. For example, nbtjobs0.sql for the NBBJOBS trigger and nbtjob20.sql for the NBRJOBS trigger.

Scripts that create packages, procedures, and functions

All scripts in the dbprocs directory that pertain to the creation of database objects that can be packages, procedures, and functions are named using the following standard.

`abcdddd.sql`

`a` = Product identifier (S)tudent, (P)ayroll, etc.

`b` = Module (E)mployee, (B)udget, etc.

`c` = Pac(K)age, (P)rocedure, (F)unction

`dddd` = Four-character mnemonic which uniquely identifies the object

Note: If the product identifier is two characters, the standard becomes `aabcddd.sql`.

The following table lists examples of this naming standard.

`gefcmnt.sql`	(G)eneral (E)vent (F)unction for (CMNT) comments
`shkgpac.sql`	(S)tudent Academic (H)istory Pac(K)age for (G)rade (P)oint (A)verage (C)alculation.
`nbkencc.sql`	Positio(N) Control (B)udget Pac(K)age for (Enc)umbrance (C)alculation.
`noforgc.sql`	Positio(N) Control (O)verall (F)unction for (Org) (C)harting.

The same 7-character name will be used to name the package object within the database.

Line extension products

Line extension products use `zzacbddd` (the `c` before the `b` is intentional).

where:

`zz`= Line extension product.

`a`= Baseline product identifier.

`c`= Pac(K)age, (P)rocedure, (F)unction

`b`= Module name.`ddd`= Table identifier

For example:

`hwpkeinf`

`hw` = Self-Service line extension product

`p` =Human Resources baseline product

`k` = Pac(K)age

`e` = (E)mployee module.

`inf` = (Inf)ormation

At the discretion of the programmer/project leader, the specification for the package may or may not, be separated from the body. They are typically separate unless they are very small packages. When split, the two scripts would be named the same except for a `1` appended to the body script name.

For example, `shkgpac.sql` would be the script to create the specification and `shkgpac1.sql` would be the script to create the body. You can use all eight characters for the specification script, for example, `sckgpac0.sql` would be the script to create the specification and `shkgpac1.sql` would be the script to create the body.

Triggers

Database trigger objects within the database are named as follows.

`at_abcdddd_xxxxxxxxxxxxxxxxxx` (a total of 29 characters)

where:

`a` = Product identifier (S)tudent, (P)ayroll etc.

`t` = (T)rigger

`abcdddd` = Table name

`xxxxxxxxxx`.... = Meaningful trigger name up to 18 characters in length

For example:

`gt_spriden_name_compress`

`pt_pebempl_audit_trail_upd`

Packages

Packages should contain their functions, procedures, etc. in alphabetical order by object name.

Procedures and functions can be created as stand-alone objects or contained within a package. There are a number of factors that contribute to this decision; therefore, it is determined by the programmer/technical project leader. The database objects that are procedures or functions will be named as follows:

`p_xxxxxxxxxxxxxxxxxxxxxxxxxxx` (a total of 29 characters)

`f_xxxxxxxxxxxxxxxxxxxxxxxxxxx`

where

`p` = (P)rocedure

`f` = (F)unction

`xxxxxxxxxx`.... = Meaningful name up to 27 characters in length

For example:

```
p_grade_point_avg_calc

f_fund_override

p_salary_enc_calc

f_check_for_event_comments
```

For Oracle to execute a SQL statement that calls a packaged function, you must assert its purity level by coding the pragma `RESTRICT_REFERENCES` directive in the package specification. The pragma `RESTRICT_REFERENCES` directive is not required to execute a packaged function in procedural statements. Please refer to the *Oracle 9i Application Developer's Guide - Fundamentals, Release 2* for more information.

Although, based on this standard, the names of functions and procedures can be up to 29 characters in length, it is strongly recommended that the names be kept shorter where possible. Many products outside of Banner have size limitations for these names; therefore, a shorter name is safer.

Cursors

Cursors are named as follows.

`xxxxxxxxxxxxxxxxxxxxxxxxxxxxC` (a total of 29 characters)

Where:

`C` = (C)ursor

`xxxxxxxxxx`.... = Meaningful name up to 28 characters in length. It is strongly recommended that the `C` be preceded by an underscore.

For example:

```
ytd_benefit_values_C

students_who_are_employees_C

delinquent_accounts_C
```

If the cursor returns all the columns for one table it is recommended that the cursor name = tablename_C (i.e., `Spbpers_C`, `Stvterm_C`).

User-defined types

User-defined types are named as follows in the database.

`a_xxxxxxxxxxxx[_nt]` (up to 29 characters)

Where:

`a` = product identifier

`xxxxxxxx...` = mnemonic that uniquely identifies the object

`_nt` literal is added to the end if the object is a nested type

For example:

`g_idname_search`

`g_idname_search_nt`

A synonym must be created for all packages for the objects within those packages to be accessed by all Oracle Tools. For example, you cannot invoke `baninst1.nbkencc.p_count_days_fisc_yr` from Oracle Forms because of the two periods (i.e., `owner.package_name.procedure_name`). A synonym must be created for the package by stripping off the BANINST1 owner. In our example we end up with a synonym named `nbkencc` and so we can then invoke the procedure by referencing it as `nbkencc.p_count_days_fisc_yr`. Using the synonym to mask the BANINST1 owner in this fashion is also consistent with how we handle the table names.

A synonym must be created for all packages. The synonym name is the same as the package except that the BANINST1 owner designation is stripped from the front.

For example:

Package name: `baninst1.nbkencc`

Synonym: `nbkencc`

Indexes

The unique index on each table is named as follows.

`7-character table name_key_index`

Each additional index is numbered numerically, starting with *2* after indexes, as follows:

`7-character table name_key_index2`

`7-character table name_key_index3`

etc.

Banner constraint naming convention

The following four constraint types are available in Oracle databases.

1. Primary key constraints — to enforce unique, non-null keys
2. Foreign key constraints — to ensure children rows are not updated/inserted if parent rows do not exist, and to prevent the deletion of parent rows if children rows do exist
3. Check constraints — to enforce integrity issues specified by the check condition
4. Unique constraints — designates a column or a combination of columns as a unique key

A constraint name must be unique for a given owner.

Note: Some foreign key constraints are delivered disabled to remove the negative performance impact of the additional indexes. They are for documentation purposes only.

Primary keys

Primary keys must be defined in the following fashion.

```
"PK_" + ppppppp
```

```
where PK for Primary Key,
```

```
ppppppp = primary key table name
```

Example: The primary key for STVTERM should be named `PK_STVTERM`.

Foreign keys

Foreign keys can be defined in the following two situations.

- Defining referential integrity constraints referencing the validation tables
- Defining referential integrity constraints for application hierarchy

Defining referential integrity constraints referencing the validation tables

Foreign keys in this category should be defined as follows.

```
"FK" + n + "_" + fffffff + "_INV_" + ppppppp + "_CODE"
```

where `FK` for Foreign Key

```
n = an one-up number to distinguish potential duplicate foreign key
 names in a given table
fffffff = foreign key table name
```

```
ppppppp = primary key table name
```

Example: The foreign key name for column `SCBCDEP_TERM_CODE_START` should be `FK1_SCBCDEP_INV_STVTERM_CODE`.

The foreign key name for column `SCBCDEP_TERM_CODE_END` should be `FK2_SCBCDEP_INV_STVTERM_CODE`.

Defining referential integrity constraints for application hierarchy

Foreign keys in this category should be defined in the following fashion.

```
“FK” + n + “_” + fffffff + “_INV_” + ppppppp + “_KEY”
where FK = Foreign Key
n = an one-up number to distinguish potential duplicate
foreign key names in a given table
fffffff = foreign key table name
ppppppp = primary key table name
```

Check constraints

The following two possible standards are recommended.

1. “CC” + n + “_” + ccccccc

   ```
   where CC for Check Constraint
   n = an one-up number to distinguish potential duplicate check
    constraint key names in a given table
   ccccccc = column name
   ```

   Example: The check constraint name for checking the range of `SCRSCHD_WORKLOAD` would be `CC1_SCRSCHD_WORKLOAD`.

2. “CC” + x + “_” + ttttttt + “_” + mmmmmmm

   ```
   where CC = Check Constraint
   x = a checking category code
   ```

   For example, R for range checking, V for value checking, etc.

   ```
   ttttttt = table name
   mmmmmmm = message
   ```

   Example: The check constraint name for checking the range of `SCRSCHD_WORKLOAD` would be `CCR_SCRSCHD_outside_0_and_999`.

Unique constraints

Unique constraints must be defined in the following fashion.

```
“uk” +n+_ppppppp +_+ ddddddd
where UK for unique constraint
```

```
n= an one-up number to distinguish potential duplicate unique
 constraints in a given table.
ddddddd= descriptive name
```

Example 1

To illustrate the situation where referential integrity is to be defined for the application hierarchy, let us assume there are three parent-child tables in the system:

```
XXXXXXX, YYYYYYY and ZZZZZZZ.
12:14:40 SQL> desc XXXXXXX;
Name                            Null?    Type
------------------------------- -------- ----
```

```
 A                                        CHAR(1)
12:14:47 SQL> desc YYYYYYY;
Name                            Null?    Type

------------------------------- -------- ----
```

```
 A                                        CHAR(1)
 B                                        CHAR(1)
12:14:51 SQL> desc ZZZZZZZ;
Name                            Null?    Type

------------------------------- -------- ----
```

```
 A                                        CHAR(1)
 B                                        CHAR(1)
 C                                        CHAR(1)
```

Table YYYYYYY is the child of table XXXXXXX and table ZZZZZZZ is the child of table YYYYYYY.

The following statement defines primary key for table XXXXXXX to enforce a unique, not null value:

```
12:14:56 SQL> alter table XXXXXXX
12:15:03   2        add constraint PK_XXXXXXX
12:15:12   3        primary key ( A );
```

```
Table altered.
```

The following statement defines foreign key for table YYYYYYY referencing the primary key of table XXXXXXX to ensure the value of A exists in XXXXXX before allowing inserts/updates to YYYYYY. Deletes of A from XXXXXX only when the value of A does not exist in YYYYYY:

```
12:15:23 SQL> alter table YYYYYYY
12:15:28   2        add constraint FK1_YYYYYYY_INV_XXXXXXX_KEY
12:15:43   3        foreign key ( A )
12:15:51   4        references XXXXXXX ( A );
Table altered.
```

The following statement defines primary key for table YYYYYYY:

```
12:16:12 SQL> alter table YYYYYYY
12:16:18   2        add constraint PK_YYYYYYY
12:16:26   3        primary key ( A, B );
Table altered.
```

The following statement defines foreign key for table ZZZZZZZ referencing the primary key of table YYYYYYY:

```
12:16:38 SQL> alter table ZZZZZZZ
12:16:43   2        add constraint FK1_ZZZZZZZ_INV_YYYYYYY_KEY
12:16:57   3        foreign key ( A, B )
12:17:09   4        references YYYYYYY ( A, B );
Table altered.
```

Example 2

Using the sample defined above, the following error messages are generated when constraints are violated:

The following statement inserted a row into table XXXXXXX successfully:

```
12:18:03 SQL> insert into XXXXXXX values ( '1' );
1 row created.
```

The following statement failed the `PK_XXXXXXX` primary key constraint, because a primary key must have unique value:

```
12:18:14 SQL> insert into XXXXXXX values ( '1' );
insert into XXXXXXX values ( '1' )
*
ERROR at line 1:
ORA-00001: unique constraint (SATURN.PK_XXXXXXX) violated
```

The following statement passed `FK1_YYYYYYY_INV_XXXXXXX_KEY` constraint checking and added a row into table YYYYYYY;

```
12:18:26 SQL> insert into YYYYYYY values ( '1', '1' );
1 row created.
```

The following statement caused foreign key violation, because there is not a primary key value ('2', '2') in table YYYYYYY yet:

```
12:18:53 SQL> insert into ZZZZZZZ values ( '2', '2', '1');
insert into ZZZZZZZ values ( '2', '2', '1' )
*
ERROR at line 1:
ORA-02291: integrity constraint (SATURN.FK1_ZZZZZZZ_INV_YYYYYYY_KEY)
 violated - parent key not found
```

Data format recommendations

To ensure consistent information throughout your Banner system, data should be entered in a standard way. See Chapter 3, “Getting Around Banner, “ in the *Banner Getting Started Guide* for recommendations on the format of IDs, names, addresses, dates, and the use of special characters.

Delivered user IDs

Below is a list of the user IDs that are delivered with Banner.

Note: Some of the IDs are used with systems that are no longer part of the Banner suite. They will be made obsolete in a future release.

The following are sample user accounts for role-level security, etc.

USR IDs	Description
ADISUSR	Sample user account for role-level security, etc., for Advancement.
BAN_SS_USER	This user is used for pooled database connections of the Banner 9 (Self-Service) web application.
FAISUSR	Sample user account for Financial Aid.
FIMSUSR	Sample user account for Finance.
FLEXREG_USER	The database uses this user for all transactions created by Flexible Registration process.
FLEXUSR	jdbc.user identified in the efc.ear deployment.

USR IDs	Description
FTAEUSR	User account used for Travel and Expense Management.
HRISUSR	Sample user account for Human Resources.
INFMGR	Kiosk Banner product owner.
LCBMGR	User account used for Banner Luminis Channels.
SAISUSR	Sample user account for Student.

The following IDs own sample and seed data in system:

PRD IDs	Description
ADISPRD	Sample and seed data owner for Advancement.
FAISPRD	Sample and seed data owner for Financial Aid.
FIMSPRD	Sample and seed data owner for Finance.
GENLPRD	Sample and seed data owner for General.
HRISPRD	Sample and seed data owner for Human Resources.
POSNPRD	Sample and seed data owner for Position Control.
SAISPRD	Sample and seed data owner for Student.
TAISPRD	Sample and seed data owner for Accounts Receivable.

The following are schema, object owners, etc.:

Other IDs	Description
ADISDAT	Advancement data user.
ALUMNI	Advancement schema owner.
BANIMGR	Banner Document Management Suite schema owner.
BANINST1	Owner of most product packages, functions and procedures.
BANJSPROXY	This is the Oracle*Wallet proxy user account used for Banner Job Submission.
BANPROXY	User ID for Connection pooling to enable one user to authenticate as BANPROXY to share sessions instead of creating new ones.
BANSECR	Security schema owner.

Other IDs	Description
BANSSO	User ID and schema owner for Single Sign-on.
BASELINE	Special user for certain delivered data. The BASELINE ID is not delivered.
BPISMGR	OBSOLETE – Property Tax schema owner.
BPISPRD	OBSOLETE – Sample and Seed Data owner for Property Tax.
BPISUSR	OBSOLETE – Sample User for Property Tax.
BSACMGR	Banner Student Aid (Canada) schema owner.
BSACUSR	Sample user for Banner Student Aid (Canada).
BWAMGR	Advancement Self-Service schema owner.
BWFMGR	Finance Self-Service schema owner.
BWGMGR	Web General schema owner.
BWLMGR	Faculty Self-Service schema owner.
BWPMGR	Employee Self-Service schema owner.
BWRMGR	Financial Aid Self Service schema owner.
BWSMGR	Student Self-Service schema owner.
CASCADEU	Cascade user used by banner-ssb-ws application.
CIMSMGR	OBSOLETE – Courts schema owner.
CIMSPRD	OBSOLETE – Sample and Seed Data owner for Courts.
CIMSUSR	OBSOLETE – Sample User for Courts.
DBEU_OWNER	User account used for the installation and administration of the Database Extension Utility (DBEU).
DCRSMGR	OBSOLETE – Cash Receipts schema owner.
DCRSPRD	OBSOLETE – Sample and Seed Data owner for Cash Receipts.
DCRSUSR	OBSOLETE – Sample User for Cash Receipts.
EPRINT	E-print schema owner.
EWQSMGR	OBSOLETE – Electronic Work Queue schema owner.
EWQSUSR	OBSOLETE – Sample User for Electronic Work Queue.

Other IDs	Description
FAISDAT	Financial Aid data user.
FAISMGR	Financial Aid schema owner.
FIMSARC	Finance archive user.
FIMSDAT	Finance data user.
FIMSMGR	Finance schema owner.
FLEXREG	Banner Flexible Registration schema owner.
GENERAL	General schema owner.
HRISDAT	Human Resources data user.
ICMGR	Integration components schema owner.
INFMGR	Kiosk Banner product owner.
INTEGMGR	Default Oracle ID for Banner Channels.
LIMSARC	OBSOLETE – Occupational Tax and License archive user.
LIMSMGR	OBSOLETE – Occupational Tax and License schema owner.
MICROFA	Obsolete.
MICRPRD	Obsolete.
MUTREP	Mass Data Update Utility schema owner – see also PRGNREP.
NLSUSR	Integration Manager schema owner.
NOSLEEP	Used by NOSLEEP triggers to get runtime parameters.
PAYROLL	Payroll schema owner.
POSNCTL	Position Control schema owner.
PRGNREP	Process Engine schema owner – see also MUTREP.
SAISDAT	Student data owner.
SATURN	Student schema owner.
STREAMSADMIN	User account used for the administration of Streams processes.
TAISMGR	Accounts Receivable schema owner.
UIMSMGR	OBSOLETE – Utilities Customer Information System schema owner.
UIMSPRD	OBSOLETE – Sample and Seed Data owner for Utilities Customer Information System.

Other IDs	Description
UIMSUSR	OBSOLETE – Sample User for Utilities Customer Information System.
VRSMGR	Voice Response Student and Financial Aid schema owner.
WFAUTO	Automated activities for a Workflow account.
WFEVENT	Event Queue Manager for a Workflow account.
WFQUERY	Query-only Workflow account.
WTAILOR	Web Tailor schema owner.
XRISMGR	OBSOLETE – Records Indexing schema owner.
XRISPRD	OBSOLETE – Sample and Seed Data owner for Records Indexing.
XRISUSR	OBSOLETE – Sample User for Records Indexing.

To generate a list of these user IDs in Oracle, enter the following command:

```
select username from dba_users order by username;
```

For security purposes, the schema owners and BANINST1 user accounts can be locked or have their passwords changed to prevent anyone from using these accounts during regular processing.

Note: You may want to set or review the setting on the User ID Restrictions section on the GSASECR page to help ensure the Banner security of these users.

BASELINE and LOCAL User IDs

Many General tables have an assigned user ID of either BASELINE or LOCAL. This user ID column identifies deliverable rows versus your custom rows so that when Ellucian delivers software in subsequent versions, there is no impact to your custom rows.

You should not change BASELINE rows without careful consideration of your future need to maintain these rows. If you need to change BASELINE rows, you can create a user with the name of BASELINE and the class of General objects. This BASELINE user would then be able to log into Banner and make changes to the BASELINE rows.

Places where you will find this most helpful are in initial set up of the standard toolbar icons and when you need to make changes to the options in the navigation frame.

Directory structure

The directory structure.

ADMIN	
OPSYS	Contains COBOL make files for platform (UNIX, AIX, DGUX, SUNOS, etc.)

ALUMNI (Banner Advancement)	
C	Pro*C and C source files
DBPROCS	SQL*Plus scripts to recreate database procedures, packages, functions, and triggers
FORMS	Oracle*Forms .fmb, .fmx, .pll, and .lib files
INSTALL	.SCTDMP file used during the initial install (renamed to .DMP during install)
MISC	Shell scripts (UNIX only)
PLUS	SQL*Plus scripts
VIEWS	SQL*Plus scripts to recreate views

ARSYS (Banner Accounts Receivable)	
C	Pro*C and C source files
COB	Pro*COBOL files (UNIX only)
DBPROCS	SQL*Plus scripts to recreate database procedures, packages, functions and triggers
FORMS	Oracle*Forms .fmb, .fmx, .pll and .lib files, Oracle Reports
INSTALL	.SCTDMP file used during initial install (renamed to .DMP during install)
MISC	Shell scripts (UNIX only)
PLUS	SQL*Plus scripts
VIEWS	SQL*Plus scripts to recreate views

COMMON
Objects shared by all products (see Chapter 5)

FINAID (Banner Financial Aid)	
C	Pro*C and C source files

FINAID (Banner Financial Aid)	
COB	Pro*COBOL files (UNIX only)
DBPROCS	SQL*Plus scripts to recreate database procedures, packages, functions and triggers
FORMS	Oracle*Forms .fmb, .fmx, .pll and .lib files
INSTALL	.SCTDMP file used during initial install (renamed to .DMP during install)
JAVA	Files that contain Java code
MISC	Shell scripts (UNIX only)
PLUS	SQL*Plus scripts
VIEWS	SQL*Plus scripts to recreate views

FINANCE (Banner Finance)	
C	Pro*C and C source files
DBPROCS	SQL*Plus scripts to recreate database procedures, packages, functions and triggers
DESKTOP/EDI	EDI Desktop Application
FORMS	Oracle*Forms .fmb, .fmx, .pll and .lib files, Oracle Reports
INSTALL	.SCTDMP file used during initial install (renamed to .DMP during install)
MISC	Shell scripts (UNIX only)
PLUS	SQL*Plus scripts
VIEWS	SQL*Plus scripts to recreate views

GENERAL (Banner General)	
C	Pro*C and C source files, C compile procedures, EXEC INCLUDE files (source files)
COB	COBOL copybooks for all products (UNIX also includes General Pro*COBOL & .gnt files)
COB/LIB	Links to copybooks with .cob extension and lower case names (UNIX only)
DESKTOP	Desktop executable
DBPROCS	SQL*Plus scripts to recreate database procedures, packages, functions and triggers
EXE	Compiled PRO*COBOL executables for all products

GENERAL (Banner General)	
FORMS	Oracle*Forms .fmb, .fmx, .mmb (menus), .mmx, .pll (PL/SQL library) and .lib (library) files
GIF	Banner GIFs
INSTALL	.SCTDMP file used during initial install (renamed to .DMP during install)
JAVA	Files that contain Java code
LOADER	Oracle*Loader
MISC	Shell scripts (UNIX only)
PLUS	SQL*Plus scripts
VIEWS	SQL*Plus scripts to recreate views
XSD	Oracle schemas

INSTALL
All Banner installation scripts

LINKS (UNIX Only)
Composite directory for local access of Banner products

PAYROLL (Banner Payroll)	
C	Pro*C and C source files
COB	Pro*COBOL files (UNIX only)
DBPROCS	SQL*Plus scripts to recreate database procedures, packages, functions and triggers
DESKTOP	Doc files
FORMS	Oracle*Forms .fmb, .fmx, .mmb (menus), .mmx, .pll and .lib files
INSTALL	.SCTDMP file used during initial install (renamed to .DMP during install)
MISC	Shell scripts (UNIX only)
PLUS	SQL*Plus scripts
VIEWS	SQL*Plus scripts to recreate views

POSNCTL (Banner Position Control)	
C	Pro*C and C source files

POSNCTL (Banner Position Control)	
DBPROCS	SQL*Plus scripts to recreate database procedures, packages, functions and triggers
FORMS	Oracle*Forms .fmb and .fmx files
INSTALL	.SCTDMP file used during initial install (renamed to .DMP during install)
MISC	Shell scripts (UNIX only)
PLUS	SQL*Plus scripts
VIEWS	SQL*Plus scripts to recreate views

STUDENT (Banner Student)	
C	Pro*C and C source files
COB	Pro*COBOL files (UNIX only)
DBPROCS	SQL*Plus scripts to recreate database procedures, packages, functions and triggers
FORMS	Oracle*Forms .fmb, .fmx, .pll and .lib files, Oracle Reports
INSTALL	.SCTDMP file used during initial install (renamed to .DMP during install)
JAVA	Files that contain Java code
LOADER	Oracle*Loader
MISC	Shell scripts (UNIX only)
PLUS	SQL*Plus scripts
VIEWS	SQL*Plus scripts to recreate views

COBOL standards

It is difficult to fully document exactly how a Banner COBOL program is to be written. Many factors influence the particular programming approach that should be followed to satisfy specific requirements.

This section gives some guidelines and recommendations which should be followed when an existing Banner COBOL program is modified or a new one created.

These guidelines are divided into three sections: Rules, Standards, and Style. Rules should always be followed; standards should be followed unless there is a demonstrable need to do otherwise; and styles are recommendations.

In general, rules address operating system portability, ANSI compliance, and Oracle version compatibility. Standards enhance the maintainability of the code. Style relates primarily to the appearance of the COBOL source code.

Rules

This rule applies only to those programs that perform a connect to an Oracle database.

Banner COBOL programs must make use of some of the General support objects to gain access through the security routines. Two include files (also referred to as copybooks) are required, and a Working Storage variable must be initialized. Additionally, the program should be able to be compiled with the “sqlcheck= full” option. In certain circumstances however, this is not possible. For example, GLBLSEL.pco cannot be compiled in this manner at sites which do not have Financial Aid because the program references the RORVIEW TABLE. Compiling with “sqlcheck= full” in this case would result in an error.

The first required include file is SETSEED. This must be placed immediately before the EDECLARE include file, or, if EDECLARE is not used, immediately before the END DECLARE statement in Working Storage. For example:

```
EXEC SQL INCLUDE SETSEED END-EXEC.
EXEC SQL INCLUDE EDECLARE END-EXEC.
```

The variable `OBJECT-NAME` is declared in SETSEED, and must be initialized just before the include of the second include file that is required for security processing, SETROLE. The variable initialization and include statement must be placed immediately after the connect to Oracle, as shown in the example below:

```
EXEC SQL
CONNECT :USERID IDENTIFIED BY :PASSWRD
END-EXEC.
MOVE '<program name> 'TO OBJECT-NAME.
EXEC SQL INCLUDE SETROLE END-EXEC.
```

After ensuring that the above files are included, the program should be compiled with the “sqlcheck=full” option.

Comment lines between logically grouped blocks of COBOL sentences are encouraged as they make the program easier to read. Every comment line must contain an asterisk in column 7. In other words it must be officially designated as a comment line. Certain compilers yield a syntax error if they encounter a blank line that is not truly a comment line.

```
215-8 *
215-8 3800-DELETE-ALL-FROM-NHRFINC.
215-8  MOVE '3800' TO ABORT-PARA.
215-8 *
215-8  EXEC SQL
215-8  DELETE FROM NHRFINC
215-8   WHERE NHRFINC_CATEGORY_CODE BETWEEN 'A' AND 'J'
215-8     AND NHRFINC_INTERFACED_IND = 'Y'
```

```
215-8  END-EXEC.
215-8 *
215-8  EXEC SQL COMMIT WORK END-EXEC.
215-8 *
215-8 3800-EXIT.
215-8  EXIT.
```

When declaring variables in WORKING-STORAGE, the word PICTURE must be spelled out fully as opposed to using the PIC abbreviation. Some compilers do not accept the abbreviation.

```

WORKING-STORAGE SECTION.
 EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01  MISC-DECLARE-SECTION-VARIABLES.
 05 USERID PICTURE X(20).
 05 PASSWRD PICTURE X(20).
 05 CONTROL-DISP PICTURE X(02) VALUE '60'.
 05 UPDATE-DISP PICTURE X(02) VALUE '62'.
 05 WORK-DATE PICTURE X(11).
```

Literals should be enclosed in single quotes ('xxx') instead of double quotes (“xxx”). This applies to the VALUE clause in WORKING-STORAGE (as shown above) and literals used in the PROCEDURE DIVISION.

```

*========================================================
 * Added the following logic to determine if an automatic
 * login is being used, and if so, set up the field values
 * required for an automatic login.
 *========================================================
 IF EACH-PARM (2) = '/'
  MOVE '/'  TO DQY-AUTO-LOGIN-ARR
   IN DQY-AUTO-LOGIN
  MOVE 1 TO DQY-AUTO-LOGIN-LEN.
 *
```

Standards

Record all changes to a program in the Audit Trail section at the top of the program directly before the ENVIRONMENT DIVISION statement.

The Audit Trail follows a particular format which includes the sequential number of the modification within a release, a description of the change, the programmer's initials and the date.

It should be proceeded with a brief description of the purpose of the program.

```

**************************************************************
*  This is the Population Selection extract program. It will
*  create a list of PIDMs for a given selection ID, which can
*  be used as input to the Letter Generation extract, or other
*  reports.
**************************************************************
* AUDIT TRAIL: x.x
```

```
*
*  1. SJQ  05/14/1991
*     RENAME PROGRAM NAME TO UPPERCASE.
*  2. JEF 05/31/91
*     Rename ROBDATA to GLBDATA and add letter generation
*     modifications.
*
```

More recently, an alternative technique has been employed whereby the Audit Trail entry includes a Problem or Need statement, a Functional Impact and a Technical changes statement.

```

*  2. RPM # 475. RLP 01/04/96
*     Need - Computer Calculated Manual Checks should
*            default to disposition '40'.
*     Functional Impact - User no longer will have to balance
*     computer calculated events that
*     have been processed by   PHPCALC.
 *    Immediately after PHPCALC has been
 *    run, the user can    now run PHPDOCM.
*
 * Technical changes - All references to a disposition of'37’
 *    have been removed or changed to '40'.
*
*
```

Each line of code that is affected by a particular modification should contain a “Mark Mod” in columns 1 through 6 which indicates the release and sequential number of the change. For example, the first modification in the program for release 2.1.7 would be marked with 217-1 in columns 1 through 6. This is extremely useful when trying to fully track how, when and why a certain line of code was changed. The following code includes lines affected by the tenth modification of the 2.0 release and the seventh modification of the 2.1.5 release:

```

IF DQY-ERROR-TYPE = 'F'
           EXEC SQL ROLLBACK WORK END-EXEC
           MOVE DQY-ERROR-MSG TO GJBRSLT-MESSAGE
           MOVE SPACES TO DQY-ERROR-MSG,
           DQY-ERROR-TYPE
           MOVE 'F' TO GJBRSLT-STATUS-IND
20-10      PERFORM DQY-INS-GJBRSLT THRU DQY-INS-GJBRSLT-EXIT
215-7      EXEC SQL COMMIT WORK END-EXEC
            PERFORM DQY-ABORT THRU DQY-ABORT-EXIT.
```

Each program should include a display statement up front in the logic of the format Starting <program_name> (Release x.x.xx). The release number must be updated with each release for which the program is modified. This gives a clear and easy way to verify that the correct program and version of that program is being executed.

```

2000-SIGN-ON-TO-DBMS.
           MOVE  '2000' TO ABORT-PARA.
13-14      DISPLAY ' '.
217-1      DISPLAY 'Starting PHPFEXP (Release 7.1.1)'.
           DISPLAY ' '.
```

```
            DISPLAY 'Username: '.
            ACCEPT USERID.
            MOVE SPACE TO WS-LOWER-CASE
             WS-UPPER-CASE.
```

Certain versions of the COBOL compiler behave differently with respect to command line parameters and accepting data from the console (terminal). To accommodate these differences, it was necessary to provide for a “dummy ACCEPT” statement as the first ACCEPT in the program. A pre-compile definition of “SCT001” is used in conjunction with the standard Banner compile scripts to allow control of whether this dummy ACCEPT is needed or not. Only when the SCT001 parameter is defined for the pre-compiler will the dummy ACCEPT end up in the executable code. It is recommended that these first three lines of code be included in all programs.

```
10000-ENTER-PROGRAM.
20-14       EXEC ORACLE IFDEF SCT001 END-EXEC.
            ACCEPT WS-DUMMY-ITEM FROM USER-INPUT-DEVICE.
20-14       EXEC ORACLE ENDIF END-EXEC.
217-3 *
217-3       MOVE SPACES TO GJBRSLT-FIELDS.
217-3        PERFORM 11000-RETRIEVE-ONLINE-PARMS THRU 11000-EXIT.
```

Style

Care should be taken when lining up and indenting WORKING-STORAGE variable definitions. It is much easier on the eye and is more conducive to understanding the program when one does not have to struggle with confusing formatting that makes it difficult to discern the level relationships between variables.

How it should be done:

```
20-9  01 FRINGE-CHARGE-BACK-WORK-AREA.
20-9     05  FBLD-HIT-SW              PIC X(02).
20-9     05  FBLD-QUERY-DATE          PIC S9(07) COMP-3.
20-9     05  FBLD-COAS-CODE           PIC X(01).
20-9     05  IO-NTRFBIN-RATE          PIC S9(04)V999 COMP-3.
20-9     05  IO-NTRFBEX-RATE          PIC S9(04)V999 COMP-3.
202-2    05  FRINGE-POSTING-MODE      PIC  X(01).
202-2    05  FRINGE-INST-AMOUNT       PIC S9(07)V99  COMP 3.
20-9     05  IO-NTRFBEX-FOAPAL.
20-9         07  IO-NTRFBEX-FUND-CODE PIC  X(06).
20-9         07  IO-NTRFBEX-ORGN-CODE PIC  X(06).
20-9         07  IO-NTRFBEX-ACCT-CODE PIC  X(06).
20-9         07  IO-NTRFBEX-PROG-CODE PIC  X(06).
20-9         07  IO-NTRFBEX-ACTV-CODE PIC  X(06).
20-9          07  IO-NTRFBEX-LOCN-CODE   PIC  X(06).
```

How it should not be done:

```
20-9 01 FRINGE-CHARGE-BACK-WORK-AREA.
20-9 05 FBLD-HIT-SW PIC X(02).
20-9 05 FBLD-QUERY-DATE PIC S9(07) COMP-3.
```

```
20-9 05 FBLD-COAS-CODE PIC X(01).
20-9 05 IO-NTRFBIN-RATE PIC S9(04)V999 COMP-3.
20-9 05 IO-NTRFBEX-RATE PIC S9(04)V999 COMP-3.
202-2 05 FRINGE-POSTING-MODE PIC X(01).
202-2 05 FRINGE-INST-AMOUNT PIC S9(07)V99 COMP-3.
20-9 05 IO-NTRFBEX-FOAPAL.
20-9 07 IO-NTRFBEX-FUND-CODE PIC X(06).
20-9 07 IO-NTRFBEX-ORGN-CODE PIC X(06).
20-9 07 IO-NTRFBEX-ACCT-CODE PIC X(06).
20-9 07 IO-NTRFBEX-PROG-CODE PIC X(06).
20-9 07 IO-NTRFBEX-ACTV-CODE PIC X(06).
20-9 07 IO-NTRFBEX-LOCN-CODE PIC X(06).
```

The care needed for WORKING-STORAGE indentation applies similarly to the PROCEDURE DIVISION. It is best to illustrate with examples:

```

PERFORM 2000-SIGN-ON-TO-DBMS THRU 2000-EXIT.
           PERFORM 3000-GET-PARAMETERS  THRU 3000-EXIT.
215-8 *
215-8     PERFORM 4100-INITIALIZE-CAT-TOTALS THRU 4100-EXIT
215-8     VARYING CAT-SUB FROM 1 BY 1
215-8     UNTIL CAT-SUB > CAT-MAX.
215-8 *
215-8      MOVE ZEROS TO LIQUIDATION-TOTAL-D
215-8      LIQUIDATION-TOTAL-C.
215-8 *
215-8      IF PARM-ALL-PAYROLLS  =  'Y'
215-8      IF PARM-PICT-CODE  =  SPACES
215-8      PERFORM 3410-DECLARE-AND-OPEN-PAYS-1 THRU 3410-EXIT
215-8      ELSE
215-8      PERFORM 3420-DECLARE-AND-OPEN-PAYS-2 THRU 3420-EXIT
.
.
.
215-7      WHERE HRHIST_PAYNO = PHRJOBS_PAYNO
215-7        AND PHRHIST_PIDM = PHRJOBS_PIDM
215-8        AND ((:PARM-REDIST-ONLY = 'N')OR
215-8        (PHRHIST_TYPE_IND =  'R') OR
215-8        (PHRHIST_TYPE_IND = 'V' AND
215-8        'R' =
215-8        (SELECT PHRHIST_TYPE_IND
215-8           FROM PHRHIST Y
215-8           WHERE Y.PHRHIST_YEAR = X.PHRHIST_YEAR
215-8           AND Y.PHRHIST_PAYNO = X.PHRHIST_PAYNO
215-8            AND Y.PHRHIST_PIDM = X.PHRHIST_PIDM))
```

Paragraph names should use a numbering scheme that communicates the structural hierarchy of the PROCEDURE DIVISION logic. For example, all initial housekeeping paragraphs might be grouped in the 1000- to 1900- range. The parameter input logic might be grouped in the 2000- to 2900- range and so on. A structure based on letters can also be used (AAA1-, AAA2-, ABB1- etc.)

All paragraphs should have an exit and the perform of a paragraph should always be “THRU” the exit.

```

       3000-INITIALIZATION-CONTROL.
```

```
             PERFORM  4100-INITIALIZE-CAT-TOTALS THRU 4100-EXIT.
       3000-EXIT.
            EXIT.       4100-INITIALIZE-CAT-TOTALS.
            MOVE ZEROS TO CATEGORY-TOTAL-1,
       CATEGORY-TOTAL-2.
.
.
.
      4100-EXIT.
            EXIT.
```

C Standards

This information provides some recommendations for C code developed as part of the Banner system.

These guidelines are divided into three sections: Rules, Standards, and Style. The differences between the three are as follows: rules should always be followed; standards should be followed unless there is a demonstrable need to do otherwise; and styles are recommendations which an individual programmer may choose to discard.

The Banner C coding standards are evolving and subject to change. Most Banner Pro*C code originated as Oracle SQL*Report code that was converted to Pro*C using an automated process, and as a result may not conform to all of the rules and standards in this section; particularly, this code is rife with the goto statement. There was subsequent code developed over the years as C standards were evolving, and so not every program delivered meets the rules and standards below.

Rules

In general, items dealing with operating system portability, ANSI compliance, Oracle version compatibility, and avoidance of common errors will be treated as rules.

Procedure

1. Adherence to the ANSI C standard is paramount; any exceptions are noted below. A copy of The C Programming Language, 2nd Edition, by Brian Kernighan and Dennis Ritchie, should be standard equipment for any programmer writing or modifying Banner C code.
2. All variable declarations global to the current compilation unit, function declarations, and function prototypes must include the storage class modifier static unless they need to be available for external linkage. Global variables and function declarations are, by default, external. To support proper modularity, each program unit should only make external those functions and variables which have been determined to be necessary for other code units to access.

   ```
   /* global variables with external visibility */
   char username_password[62];
   unsigned int status_code;
   /* global variables visible only in current compilation unit */
   ```

```
static FILE *infile,*outfile;
static long line_count=0;
```

3. All functions must be fully prototyped, following ANSI standards, either in a header file (if accessed by more than one source file) or at the top of the source file where it is declared. A complete prototype consists of the return type of the function, the function name, and the types of each parameter, along with the formal parameter names. As a matter of style, the prototype should exactly match the actual function declaration, e.g.:

```
char *str2lc(char *str);
.
.
.
char *str2lc(char *str)
{
...
}
```

4. The goto statement should never be used in new C code and should be removed from all existing code when possible; this also eliminates any need for labels. Use structured programming techniques instead.

```
/* parameter validation code with gotos and labels */
askparms:
   input(ask_p_owner,"TABLE CREATOR: ",30,ALPHA);
   if ( !*ask_p_owner ) goto rdowner;
   strcpy(p_owner,ask_p_owner);
   goto nexta;
rdowner:
   strcpy(parm_no,"01");
   sel_optional_ind(FIRST_ROW);
   if ( compare(rpt_optional_ind,"O",EQS) ) goto nexta;
   goto missing_parms;
nexta:
/* parameter validation code without gotos and labels */
input(ask_p_owner,"TABLE CREATOR: ",30,ALPHA);
if ( !*ask_p_owner )
  {
    strcpy(parm_no,"01");
    sel_optional_ind(FIRST_ROW);
    if ( compare(rpt_optional_ind,"O",NES)
    missing_required_parm("Table Creator");
   }
```

5. All programs should include guastdf.h, and only this file should include standard headers. This will limit the number of changes necessary for new compilers or hardware platforms and will insure that all necessary headers are included. Note that guarpfe.h includes guastdf.h, so an explicit inclusion is not necessary.

```
#include "guarpfe.h" /* good; gets all required headers */ #include
"myheader.h" /* good; local header */ #include <sys/strtty.h> /* bad;
non-portable, system-specific */
```

6. No compiler or platform specific functions may be used. Only those functions found in the ANSI standard libraries are available on all supported platforms. Exceptions to this rule, such as

the use of the UNIX and OpenVMS provided function sleep, may be made with management approval. Refer to The C Programming Language, 2nd Edition, by Brian Kernighan and Dennis Ritchie, for a definitive listing of the functions available.

7. The following ANSI features are not available under otherwise compliant compilers, such as older versions of DEC C, and are not to be used unless all supported compilers implement them in the future: the atexit and memmove functions, concatenation of adjacent string literals, and ## macro expansion. Also, an assignment followed by the “address of” or “dereference” operator with no intervening space, is misinterpreted by some releases of DEC C. Accordingly, use

   ```
   a= *b;
   ```

   instead of

   ```
   a=*b;
   ```

   Finally, DEC C requires that main be of type `int` and return a value to the operating system.

   ```
   int main(int argc,char *argv[]) /* suggested portable declaration of
   main */
   ```

8. All handling of file names from the operating system is done with the `makefn` and `parsfn` functions defined in guastdf.h to provide maximum code portability.

   ```
   FNSTRUC outfile;
   .
   .
   .
   strcpy(outfile.fname,*argv);
   parsfn(&outfile);
   strcpy(outfile.ext,"lis");
   makefn(&outfile);
   ```

9. All programs should use the function `exit2os`, defined in guastdf.h, to return to the operating system; this will ensure that all necessary database and memory cleanup is performed. Application code should never use the standard function exit. Also, application code should never reach a return from within the main function; however, to prevent warnings from some compilers, the `exit2os` call at the bottom of main should be immediately followed by a return.

   ```
   int main(int argc,char *argv[])
   {
      ...
      /* all done */
      exit2os(EXIT_SUCCESS);
      return 0;
   }
   ```

10. All Pro*C programs must include the file guaorac.c and use the provided database utility functions for database connection and disconnection, and use the macro `POSTORA` to check for database errors. This will insulate code from future changes to Pro*C internals and provide a common interface to the database for all programs. Most importantly, the `login` function must be used to connect to the database with Banner security enabled.

   ```
   /* minimal.pc */
   ```

```
#include <guastdf.h>
EXEC SQL INCLUDE guaorac.c;
int main(int argc,char *argv[])
{
   CHAR31 myname;
   /* necessary for security in absence of rptopen */
   getxnam(*argv);
   /* login to database with three tries */
   login();
   EXEC SQL SELECT user INTO :myname FROM dual;
   /* check for error */
   POSTORA;
   printf("Logged on to Oracle as %s\n",myname);
   /* does database exit, other cleanup */
   exit2os(EXIT_SUCCESS);
  return EXIT_SUCCESS;
}
```

11. All application and support code should Pro*C pre-compile and C compile with no warnings or errors on all supported platforms. Following ANSI standards eliminates the majority of problems, but certain compilers may be more restrictive than others. For example, when using the Pro*C pseudo-datatype VARCHAR, it is necessary to explicitly typecast the arr member to a character pointer in standard function calls under some compilers. In short, when in doubt, cast.
12. All Pro*C programs should recognize the -t command-line switch to turn on the SQL trace facility. Programs converted from earlier Oracle SQL*Report code use rptopen to handle this option.

```

int main(int argc,char *argv[])
{
   extern short sqltrace_flag;
   rptopen(user_pass,argc,argv);
   login();
   if ( sqltrace_flag )
   EXEC SQL ALTER SESSION SET SQL_TRACE TRUE;
```

13. Many C compilers allow modification of literals by means of pointers; this is not allowed in our C code. Consider the following code:

    `...char *ptr="SCT"; *ptr = '\0';...`

    Here ptr points to an area of storage containing the string literal "SCT", which may not be unique storage if the same literal appears elsewhere in the program. Modifying the storage pointed to by `ptr` may work as expected, but if the new value assigned to `ptr` is longer than the original literal area, then memory errors will occur. Also, some compilers will signal an error or warning message if such an operation is attempted.
14. Always check error status after any I/O or database operation. The guastdf.h include file defines the macro `POSTORA` to make Oracle error checking simpler, and all file I/O operations should be followed by a check using the ferror standard function.

```

EXEC SQL SELECT ename
           INTO :ename:ename_ind
           FROM emp
           WHERE empno = :empno;
```

```
POSTORA;
fprintf(outfile,"%d:%s\n",empno,ename);
if ( ferror(outfile) )
   prtmsg(IOERROR,outfile_name);
```

15. Functions which return a pointer to a local variable must give the static storage class to the return variable. If the keyword static is not supplied, then the storage for the local variable may be reused by the program before the calling function is able to access the address. This is a common error which is rarely caught by the compiler or tools such as lint, and may even work correctly on some machines, depending on the way that the memory heap is managed.

    For example, consider a function that generates a new password string and returns a pointer to the new value. The calling function will then copy this value elsewhere for storage, as the value will be lost when the password function is next called.

```
char *getpassword(void)
{
   static char passval[9];
   .
   .
   .
   return passval;
}
```

16. Indicator variables must be used on all SQL output variables, and on all non-string input variables (unless a non-NULL value is guaranteed.) This is to prevent truncation warnings when the target is too small for the source, and to properly handle NULL values. See Rule 14 for an example.
17. Complex structures which will be reused should be typedefed in to simplify and clarify the code. For example, consider the structure and declarations for implementing a linked list of file information:

```
typedef struct fn_node_struct
       {char *fname;
        char *owner;
        unsigned long fsize;
        struct fn_node_struct *next_fn_node;} FN_NODE;
FN_NODE *head,*tail;
```

18. Use Oracle datatype equivalencing to handle C-style null-terminated strings in preference to the `VARCHAR` pseudo-datatype. All C code originally converted from Oracle SQL*Report code uses this method, as does most subsequent Banner code uses this method. The include files guastdf.h and guaorac.c provide predefined typedefs for string sizes from 2 to 256 characters in length (1 to 255 usable characters plus the terminating null;) if a particular application requires longer strings, or strings embedded within arrays, then use explicit Pro*C `TYPE IS` and `VAR IS` logic (see the Programmer's Guide to the Oracle Pro*C Precompiler for details).

```
/* Without datatype equivalencing */
VARCHAR zipcode[11];
.
.
.
```

```
EXEC SQL SELECT zipcode
           INTO :zipcode:zip_ind
           FROM address
          WHERE empno = :empno;
POSTORA;
zipcode.arr[zipcode.len] = '\0';
/* With datatype equivalencing */
CHAR11 zipcode;
.
.
.
EXEC SQL SELECT zipcode
           INTO :zipcode:zip_ind
             FROM address
             WHERE empno = :empno;
```

19. Use the appropriate numeric datatype for the application, keeping in mind the limitations of each. The basic choices are a C integer type, a C floating-point type, or the provided pseudo-datatype of NUMSTR.

    Integers are limited to whole numbers only, and in comparison to the Oracle internal NUMBER datatype have a small number of significant digits. A C integer datatype (e.g., long, unsigned int) should only be used as a SQL input/output variable if the Oracle column is a whole number that will never be larger/smaller than the ANSI-defined range for the C datatype. For example, the ANSI-defined minimal magnitudes for a long datatype are -231 to 231-1 (approximately +/- two billion). Integer data types may be appropriate for database columns such as counters and sequences.

    Floating-point numbers in C have a minimum of 10 significant digits in the ANSI standard. This limitation makes them inappropriate for most currency calculations. However, all Banner-supported platforms currently have at least 15 digits of precision for the double datatype, so using double as an SQL input-output variable is acceptable provided that the database column in question is known to never exceed 15 digits. This precision should be adequate for nearly all calculations involving U.S. currency, but may be inadequate for non-U.S. currency transactions.

    All C codeconverted from Oracle SQL*Report code, and much code written subsequent to the conversion, uses the NUMSTR pseudo-datatype to provide a guaranteed 24 digits of precision. This datatype is implemented by representing numbers as fixed character strings, and only the four basic arithmetic operators are provided; more elaborate calculations must be performed in the database. The advantages of this datatype are the increased precision, and the elimination of the need for indicator-variable processing (since empty strings are interpreted by Oracle as NULLs.)

Standards

Those items whose primary purpose is to enhance maintainability of the code will be standards.

Procedure

1. A consistent system for naming variables may be mandated by individual technical managers (e.g., so-called Hungarian notation). If a system is not used, then at minimum variable and function names should be long enough to be descriptive, but not so long as to interfere with a clear understanding of the code.

2. Pro*C programs which were converted from Oracle SQL*Report code have all variables created as globals within the compilation unit, a required strategy because SQL*Report provided only global variables. With new code, or modifications to converted code, good structured programming techniques dictate the usage of local variables as a general rule, with global variables reserved for those occasions when they are necessary to prevent excessively complex or awkward code.
3. If a function in one of the support files (such as guastdf.h) is available to perform the task at hand, use it instead of creating a new one. Likewise, if a function is developed which is of general utility (such as string or number handling, I/O functions, etc.) then it should be placed in one of the support files to be available for all Banner code.
4. Functions, macros, etc. which extend the language (i.e., support code such as that found in guarpfe.h or guastdf.h) should be named mnemonically, without regard to product. For example, the function to replace a string with its lower case equivalent is named str2lc. The name of any other program object which will be used by multiple programs within a specific product should be named following usual Banner rules for the initial character. For example, a Finance function to calculate available balance could be named `favlbal`.
5. Do not depend on the numeric value of a particular macro remaining unchanged or always testing to true or false. Instead, compare the value in question with the current value of the macro. There are exceptions, such as the TRUE and FALSE macros in guastdf.h, where the numeric value will always remain unchanged.

```
#define OS_VMS  0
#define OS_UNIX 1
#define OS_NT   2
.
.
.
/* don't do this */
if ( opsys )
  printf("Operating system is UNIX or Windows NT\n");
/* do this instead */
if ( opsys==OS_UNIX || opsys==OS_NT )
  printf("Operating system is UNIX or Windows NT\n");
```

6. The general structure of a C program should be as follows:

```
#include ...                  /* header file includes */
EXEC SQL INCLUDE ...          /* Pro*C includes */
#define ...                   /* macro definitions */
static void my_fcn(void);     /* function prototypes */
static int flag;              /* non-ORACLE globals */
EXEC SQL BEGIN DECLARE SECTION/* ORACLE globals */
int main(...                  /* the function main */
static void my_fcn(void)        /* all other functions */
```

Functions should be defined in some logical order, such as alphabetic or by purpose.

7. Variables which are initialized at declaration time should appear on separate lines; e.g.:

   `static int flag=TRUE,error=FALSE;`

   should be written as:

   `static int flag=TRUE, error=FALSE;`
8. Every function or other major block of code (blocks of prototypes, variable declarations, etc.) should be preceded by explanatory comments.
9. Names of macros and type definitions should usually be in all capitals to clearly differentiate them from functions and standard C features.
10. Procedural macros should not include a closing semicolon, which should instead be coded when the macro is invoked. Many programmers code macros which are not enclosed in braces with a closing semicolon, but the resulting invocation can look confusing, as a line of code without the semicolon looks “incomplete” when scanning the code.

```
#define POSTORA if ( sqlca.sqlcode < 0 ) dberror(__FILE__,__LINE__)
.
.
.
POSTORA;
```

11. Explicit SQL cursors should be closed when they are no longer needed. This may be very difficult to ascertain for converted code, but new development should follow this standard.
12. Cursor names should be descriptive. The generated Pro*C programs use one-up numeric cursor names, but new programs should be more clear.

```
...
EXEC SQL DECLARE retrieve_name CURSOR FOR
SELECT ename
FROM emp;
```

13. Consider the use of array fetches to improve the performance of programs which retrieve a large number of rows from the database. Refer to the Programmer's Guide to the Oracle Pro*C Precompiler for details.
14. All messaging for any functions added to the support code files should be handled by prtmsg, with the actual message text added to guaerror.h.
15. Keep it simple. It is easy to write cryptic, code that cannot be maintained in C; however, other programmers may need to maintain your code in the future.
16. Again, keep it simple, but not too simple. Become familiar with common C constructs and the functions available through the standard libraries. For example, here are two versions of a function that changes all occurrences of one character in a string to another character. Both functions provide correct output, but the second is “better” because it uses standard C functions and conventions to accomplish the task.

```
/* "Bad" version of chgchar */
char *chgchar(char *str,char oldc,char newc)
{
   int i;
   if ( strcmp(str,"\0") == 0 )
     printf("String is empty\n");
   else
     for ( i=0 ; str[i] != '\0' ; i++ )
       if ( str[i] == oldc )
          str[i] = newc;
   return str;
}
/* "Good" version of chgchar */
char *chgchar(char *str,char oldc,char newc)
{
   char *p=str-1;
   if ( !*str )
     printf("String is empty\n");
   else
     while ( (p=strchr(p+1,oldc)) != NULL )
        *p = newc;
   return str;
}
```

17. The SQL DECLARE SECTION syntax is now optional. With Pro*C 2.x and above all C code is parsed, so variables declared anywhere in the program, including standard declarations, function parameters, and macro expansions, are available for use as both input and output variables in SQL code. Because it is still a good idea to group variables by function, existing code may continue to use a declare section.

Style

Style relates primarily to the appearance of the C source code, and the guidelines given here describe one programmer's approach to this issue; the goal with the style guidelines is not to be prescriptive, but rather to provide guidance for novice C programmers.

Procedure

1. Be consistent; whatever style for commenting, indentation, etc., is used in a program, use the style consistently throughout the program.
2. Wherever any purely stylistic guideline interferes with the readability of the code, ignore it. The only purpose of a programming style is to enhance, not diminish, the maintainability of the program.
3. Begin functions at the left margin, with the opening and closing braces flush against the margin. All code is indented two spaces. Subsidiary blocks of code, such as targets of if statements, are likewise indented two spaces.

```
char *str2lc(char *str)
{
   char *p;
   for ( p=str ; *p ; p++ )
     if (isupper(*p))
        *p = tolower(*p);
   return str;
}
```

4. Where a block of code rather than a single line is used, the opening and closing braces should be lined up in the column for the current indentation, with the contained code indented another two spaces.

```
if (flag)
  {
    flag=FALSE;|
    if ( str )
      {
        puts(str);
        str = NULL;
      }
  }
```

5. Comment thoroughly. Use line comments where applicable (e.g., explaining a variable declaration) and block comments elsewhere. Start block comments with the comment open symbol, one space, and then the first line of the comment. End block comments with a carriage return and a comment close, lined up underneath the comment open. Block comments are also delineated by single blank lines before and after the block.

```
static char *name;  /* example line comment */
/* Here is an example of a block comment, defined as a comment which
 is longer than a single line.
*/
```

6. Avoid extraneous braces in code that is the target of an if or else statement. For example, do not code the following:

```
if (flag)
  {
    puts("TRUE");
  }
else
  {
    puts("FALSE");
  }
Instead, code the following:
if (flag)
  puts("TRUE");
else
  puts("FALSE");
```

7. For complex data structures and type definitions, indent individual members consistently for maximum readability.

```
typedef struct source_struct {char *srcline; struct source_struct
*next_source;} SOURCE;
```

8. For if and other logical statements, when the statement will not fit entirely on one line, break it and indent past the opening parenthesis.

```
for ( p=head_token ; p && p->type=RPTKEY ; p=p->next_token )
printf("%s\n",p->str);
```

Online Internal Processing

Global variables

A global variable can store a character string value of any length. Global variables can be used to store data values outside the blocks of a page, especially to pass information from one page to another when one page calls another page.

Global variables do not have to be explicitly declared or defined;they are commonly established with a setGlobal() method call to assign a value. For example, the following command assigns the value of N to the global variable INITF.

```
setGlobal("INITF", toStr("N");
```

Global variables remain defined for the duration of a Banner 9 administrative runtime session, or until the removeGlobal() method removes them.

The `general/common/BannerMain` page establishes the General global variables along with any product-specific globals for each product currently installed. The following are some of the General global variables that GUAINIT establishes:

`GLOBAL.CURRENT_DATE`	Current Date. Default is `TO_CHAR(SYSDATE,'DD-MON-YYYY').`
`GLOBAL.CURRENT_TIME`	Current Time. Default is `TO_CHAR(SYSDATE, 'HH24:MI:SS').`
`GLOBAL.CURRENT_USER`	Current User. Default is USER system variable.
`GLOBAL.HELP_CALL_FORM`	Help Call page. Default is GUAHELP.
`GLOBAL.HOSTCMD`	Host Commands. Host commands to be submitted. (Only valid in character mode.)

The following global variable is established when exiting a validation page by using the "exit with value" option:

`GLOBAL.VALUE`	Value. Value of the Validation Table Code returned by the Exit with Value key.

General global variables

The global variables for Banner General are stored in `general/common/BannerMain`. This is the page that is triggered whenever a user starts Banner. The global variables used in the session are unique for that user.

How PIDMs and IDs are generated

A PIDM (person identification master) is Banner's unique identifier for a person (or non-person entity) known to the system. For data integrity, it is important that the one-to-one correspondence between PIDMs and persons is maintained.

Banner generated new PIDMs and other IDs with the SOBSEQN table and its associated routines. When a new PIDM was needed, Banner selected and updated the current number from SOBSEQN to get the next available number. This approach, originally designed to handle PIDMs, has been extended to accommodate other types of IDs and transactions.

The introduction of the Banner Messaging Gateway application, which processes incoming messages and calls multiple Banner APIs within a single Oracle transaction, increased the likelihood that the SOBSEQN table would be locked (in the middle of generating a PIDM for another user) when needed. The locking problem resulted in the failure to produce synchronization messages from a message-enabled page.

In Release 7.0, the SOBSEQN method of incrementing PIDMs and IDs was replaced by another method to accomplish the same function. A new Identification API handles inserting new IDs and PIDMs into the SPRIDEN table. The Identification API uses Oracle sequences to determine the next available number for the ID or PIDM.

This approach eliminated the locking contention problem and greatly improved system performance. When a sequence is defined, it can be accessed and incremented by multiple users with no waiting. The Oracle sequence does not need to complete the previous transaction before the sequence can be incremented again. This allows for nearly simultaneous transactions for all users.

`ID_SEQUENCE` is the Oracle sequence that generates unique identification numbers such as `SPRIDEN_ID`. `PIDM_SEQUENCE` generates unique internal identification numbers such as `SPRIDEN_PIDM`. Two scripts, `gos_id_seq.sql` and `gos_pidm_seq.sql`, create the new Oracle sequences.

Note that using sequence generators may cause gaps in the sequence if an application selects .NEXTVAL and subsequently fails to store it.

The *Oracle Application Developer's Guide* explains how to manage sequences.

To select the next value from the sequence and increment it you can:

```
Select PIDM_SEQUENCE.NEXTVAL from dual;
```

To examine the next value without incrementing it:

```
Select PIDM_SEQUENCE.CURRVAL from dual;
```

You must drop and recreate the sequence to change the starting number.

During the 7.0 upgrade process, the current values on the SOBSEQN table will be used as the initial settings for the new sequences.

There are two functions, `F_GENERATE_ID` and `F_GENERATE_PIDM`, in the `GB_COMMON` package, to manage generating new IDs and PIDMs. All pages and processes that create new SPRIDEN records call the `P_CREATE` procedure in the `GB_IDENTIFICATION` package. `P_CREATE` in turn calls `F_GENERATE_ID` or `F_GENERATE_PIDM` as needed.

`F_GENERATE_PIDM`, when called, will select the next number from `PIDM_SEQUENCE`, then check to see if that PIDM is already in use in SPRIDEN. If it is, it continues to select the next number until it reaches a number not in use. This self-corrects for any discrepancies between the sequence's next available number is and what is actually stored in SPRIDEN. Therefore, using `F_GENERATE_PIDM` will eliminate the *Duplicate Generated PIDM* error.

Filling gaps in PIDM or ID number series

When some schools initially bring up Banner, they assign historical records to a series of ID numbers, for example, 1 through 200000. Then, they reset the sequence numbers to some higher number, for example, 1000000, so old records are easily identified by the ID number range, and a gap exists between the highest old number and the lowest new number.

About this task

It is possible to use the new sequence to fill in any historical gaps left in a PIDM or ID number series. For example, to fill gaps in a series of ID numbers:

Procedure

1. Drop ID_SEQUENCE.
2. Recreate ID_SEQUENCE with a starting number of 1
3. Create the next SPRIDEN record using any Banner 9 application.

   All Banner applications call the `F_GENERATE_ID` function and `F_GENERATE_PIDM,` when a new person record is created. The function will run its sequence generator up through all the existing numbers until it encounters the gap and return a valid unused number. This may take some time on the very first record created, but after that the system will continue incrementing the numbers and filling in any gaps.

Banner libraries

Banner libraries contains procedures used in Banner products.

GOQOLIB

The `general/common/libraries/Goqolib` contains procedures used in multiple pages across the Banner products. It is used as a library repository to store referenced triggers, blocks, windows, canvases, visual attributes, and items.

These procedures used in conjunction with the `general/common/libraries/Goqrpls` library contain the building blocks for the Banner system. The procedures are referenced into the source code and become part of the programs. Changes made to the `general/common/libraries/Goqolib` will be applied to all programs when they are regenerated.

Banner uses Referenced Procedures so that commonly executed logic can be maintained in one location rather than be repeated in multiple pages. Procedures that are used by multiple Banner systems are found in the library and are listed below.

GOQRPLS

The `general/common/libraries/Goqrpls/GoqrplsServices.java` source library includes many methods.

Name	Function
`gAddToPersonalMenu()`	Adds the current page to a user's personal menu.
`gB2kWinHelp()`	This package determines whether help exists and how to display it.
`gBlockExists()`	Checks if a block exists.
`gBtnPressed()`	Executes built-in subprogram associated with appropriate button.
`gBuildFullName()`	Builds name to support ID field validation.
`gButtonProc()`	This is a generic button procedure. It reads the NAME of the button and performs a `executeAction();.`
`gCheckAccess()`	This is a new function to check whether a user is authorized to access a program/process through job submission.
`gCheckFailure()`	Procedure that checks for page success.
`gCheckIfDupPidm()`	Checks for duplicated PIDM.

Name	Function
`gCheckQueryMode()`	Procedure that sets global to 1 if the page is in query mode; else 0.
`gCheckStatusQuery()`	Used to check whether the most recently executed built-in has succeeded (`COMMIT_FORM OR POST`).
`gCityStateNatn()`	Defaults city, state, nation, and country codes when you enter a ZIP/PC code.
`gCityStateNatn2()`	This function is similar to above function but it also set the Global.Zip value for subsequent call to the GTVZIPC page.
`gCityStateNatn3()`	Defaults city, state, nation, and country codes when you enter a ZIP/PC code and city.
`gCheckValue()`	Procedure that checks passed string for null values.
`gCOompressWorkName()`	Returns a compressed name field in all uppercase without spaces or punctuation except for the '%', which allows the field to be used in queries. Function can be passed any character field.
`gConvertEthnicityCode()`	Supports race/ethnicity processing.
`gCopyFldAttr()`	Procedure which copies a field's X and Y coordinates to globals.
`gCreateMetadata()`	Retrieves the current window's title, page name, and release number.
`gDataExtract()`	Extracts data from a page.
`gDateCallGuacaln()`	Supports entering date data from the calendar.
`gDateNextItem()`	Retrieves the next item for a date field.
`gDatePostItem()`	Used by date fields which require `gDateReformat();` function to insure proper date validation.
`gDateReformat()`	Reformatting date.
`gDateWhenNewItem()`	Used by date fields which require `gDateReformat();` function.
`gDeceasedWarning()`	Pops the warning alert for a deceased person.
`gDefView()`	Sets up the view for pop-up window.
`gDetermineCursorLocation()`	Used in multi-window pages to locate the cursor.
`gDetermineEraseGlobal()`	Erases any globals created by the `gDetermineCursorLocation();` procedure.

Name	Function
`gDetermineWinNotPrevActv()`	Used in multi-window pages where window to window navigation has no restrictions. This function is called from procedure `gDetermineCursorLocation();`.
`gDisplayAboutBanner()`	Displays the GUAABOT page when selected from the Tools menu.
`gDisplay_Alert()`	Generic call to display an alert window.
`gDisplayErrMsg()`	Displays errors passed back from database routines.
`gDisplayImage()`	Displays a stored image file associated with an ID through the GUAIMGE page.
`gDisplayLov()`	Displays appropriate List Of Values window for the current field and allows the return of the selected value to the calling field.
`gDoNewMessagesExist()`	Procedure to check the message table and display a message if the user received a new message from the last time they were notified.
`gDoWinActivated()`	Determine whether or not to execute the remainder of the logic in the when-window-activated trigger based on a page-specific global variable.
`gDuplicatePidm()`	Checks for duplicate PIDM.
`gEnvIsCharomde()`	This function returns TRUE in a non-GUI, character-mode environment.
`gEnvIsGui()`	This function returns TRUE in a Graphical User Interface environment.
`gEnvIsMac()`	This function returns TRUE in a Macintosh environment.
`gEnvIsMotif()`	This function returns TRUE in a MOTIF environment.
`gEnvIsWeb()`	This function returns TRUE in a Internet-native environment.
`gEnv_IsWebUnix()`	This function returns TRUE in a UNIX Internet-native environment.
`gEnvIsWindows()`	This function returns TRUE in a Windows environment.
`gEnvIsWindows3x()`	This function returns TRUE in a Windows 3.x environment. This will be made obsolete in a future release.

Name	Function
`gEnvIsWindows95()`	This function returns TRUE in a Windows95 environment.
`gEnvIsWindows98()`	This function returns TRUE in a Windows98 environment.
`gEnvIsWindows9x()`	This function returns TRUE in a Windows95 or Windows98 environment.
`gEnvIsWindowsNt()`	This function returns TRUE in a Windows NT environment.
`gErrors()`	This function populates public variables.
`gF5Navigation()`	Offers navigation options when F5 key is pressed.
`gFindWindowId()`	This function returns the ID of the current event's window.
`gFormsNls()`	Package supports international date formats.
`gFormShutdown()`	This procedure contains the common commands to be executed at page shutdown.
`gForemStartup()`	This procedure contains the common commands to be executed at page startup.
`gFuncBaseInfo()`	This procedure is called within the General Product Events Module pages (GEATASK, GEAPART, GEAFCOM) to bring up a window of base function information from GEAFUNC.
`gGetMainWindowTitle()`	Retrieves the title of the main window.
`gGetPipeMessages()`	This procedure checks for Electronic Approvals messages through the use of a dbms pipe named as the Oracle username. It alerts the user to how many transactions they have pending.
`gGetRwAttributes()`	Determines attributes of the root window.
`gGetSetLocalDir()`	Used in Job Submission and Graphics modules to define a user's operating system profile, including their default local directory.
`gGetUprfBurronColor()`	Checks user preferences for button color.
`gGetUprfCanvasColor()`	Checks user preferences for the page canvas color.
`gGetUprfCmForms()`	Checks user preferences for common matching.
`gGetUprfCodePromptColor()`	Checks user preferences for the code prompt color.

Name	Function
`gGetUprfConfAlert()`	Checks user preferences for alerts that information is confidential.
`gGetUprfDataextract()`	Checks user preferences for data extract routines.
`gGetUprfDeadAlert()`	Checks user preferences for alerts that a person is deceased.
`gGetUptrfDeMimeType()`	Checks user preferences for the type of file to be created in the data extract process.
`gGetUprfDePrompts()`	Checks user preferences for whether to include column headings in data extract files.
`gGetUprfDupSsnAlert()`	Checks user preferences for whether or not to display an alert for a duplicate Social Security Number.
`gGetUprfExitAlert()`	Checks user preferences for a prompt before exiting Banner.
`gGetUprfHelp()`	Checks user preferences for the location of online help.
`gGetUprfImageDir()`	Checks user preferences for the location of images.
`gGetUprfLinksCanvasColor()`	Checks user preferences for the canvas color of links.
`gGetprfLinksDesc1()`	Checks user preferences for the text of “My Links” item 1.
`gGETUprfLinksDesc2()`	Checks user preferences for the text of “My Links” item 2.
`gGetUprfLinksDesc3()`	Checks user preferences for the text of “My Links” item 3.
`gGetUprfLinksDesc4()`	Checks user preferences for the text of “My Links” item 4.
`gGetUprfLinksDesc5()`	Checks user preferences for the text of “My Links” item 5.
`gGetUprfLinksDesc6()`	Checks user preferences for the text of “My Links” item 6.
`gGetUprfLinksMyInst()`	Checks user preferences for “My Institution” link.
`gGetUprfLinksProg1()`	Checks user preferences for the URL or destination of “My Links” item 1.
`gGetUprfLinksProg2()`	Checks user preferences for the URL or destination of “My Links” item 2.

Name	Function
`gGetUprfLinksProg3()`	Checks user preferences for the URL or destination of “My Links” item 3.
`gGetUprfLinksProg4()`	Checks user preferences for the URL or destination of “My Links” item 3.
`gGetUprfLinksProg5()`	Checks user preferences for the URL or destination of “My Links” item 4.
`gGetUprfLinksProg6()`	Checks user preferences for the URL or destination of “My Links” item 6.
`gGetUprfMSGCanvaColor()`	Checks user preferences for the canvas color of the broadcast message window of the main menu.
`gGetUprfPromptColor()`	Checks user preferences for the color of popup windows.
`gGetUprfRecordColor()`	Checks user preferences for the color of highlighted records.
`gGetUprfScrollbarColor()`	Checks user preferences for the color of scrollbars.
`gGetUprfSeparatorColor()`	Checks user preferences for the color of separators.
`gGetUprfStartupMenu()`	Checks user preferences for the default expanded menu.
`gGetUprfTreeCanvasColor()`	Checks user preferences for the canvas color of the menu tree.
`gGetUprfValue()`	Supports other user preference functions by retrieving the specific user preference value, or institutional preference value if no user preference value exists.
`gGetUprfWrbbkshelf()`	Checks user preferences for the location of the Bookshelf.
`gGetUprfWebhelp()`	Checks user preferences for the location of web help.
`gGetUprfWeboutput()`	Checks user preferences for the web server database location for database procedure execution.
`gGetUprfWebrpt()`	Checks user preferences for the location of reports on the web.
`gGetUprfWebrptService()`	Checks user preferences for the report service name for `RUN_REPORT_OBJECT`.
`gGET_WIN_PROPERTY`	This procedure returns the Height, Width, and Position of the current window.

Name	Function
`gGoqolibFuncInfoBlock()`	Displays basic function information (i.e., from GEBFUNC) on event pages (i.e., GEATASK, GEADART).
`gGoqolibKeyTrigger()`	This defines the standard key functions, such as `Key_Up` and `Key_Exit`.
`gGoqolibOptBlock()`	This defines commonly used option block procedure.
`gGoqolibPpTrigger()`	This defines commonly used pre/post form triggers.
`gGoqolibUserTrigger()`	This defines commonly used key functions.
`gGuahelp()`	Procedure to call GUAHELP.
`gGuamenuCheckSet()`	Disables the Select button and menu item when the page is called from GUAMENU.
`gHelpSetup()`	Sets global for use in GUAHELP page.
`gIdnameSearch()`	Package used for the new ID/Name search logic.
`gImgDriver()`	Supports Banner Document Management Suite (BDMS) activities invoked from within Banner.
`gInsUpdLocalDir()`	Routine to insert/update the user's profile for print destination.
`gInvalidFunctionMsg()`	Shows message for key strokes that are not valid.
`gInvokeCm()`	Checks whether the user is required to use the Common Matching (GOAMTCH) page when creating an ID, and brings up GOAMTCH if required.
`gKeyOptMenu()`	Invokes the key option list window.
`gLastTen()`	This updates the Globals used to populate the Last 10 Forms list under the Action item in the Menu Bar.
`gListValuesCall()`	This procedure calls the appropriate 'TV' validation page for the current item.
`gListValuesCopy()`	Copies the value back from 'TV' page.
`gLoadFormHeader()`	Copies the heading information.
`gMMasks()`	Determines if masking rules exist for a page.
`gMenuBar()`	Routines to set the menu settings.

Name	Function
`gMouseDoubleclick()`	Determine the type of item that the cursor is currently on and launch the appropriate action when the mouse button is double-clicked.
`gNavigationFrame()`	Package containing all of the logic for establishing and executing the options in the navigation frame.
`gNchk()`	Function which performs a null value check on a passed value.
`gNvaSetButton()`	Determines the button color.
`gNvaSetCanvas()`	Determines the canvas color.
`gNvaSetItem()`	Determines the item color.
`gNvaSetKeyBlock()`	Determines the key block color.
`gNvaSetPrompt()`	Determines the prompt color.
`gNvaSetPromptCode()`	Determines the prompt code color and style.
`gNvaSetRecord()`	Determines the highlighted record color.
`gNvaSetScrollbar()`	Determines the scrollbar color.
`gNvaSetSeparatorLine()`	Determines the separator color.
`gNvaSetWindow()`	Determines the window color.
`gPopulateAtvgoftLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopulateEthnicityList()`	Populates the list of ethnicity codes.
`gPopulateFtvacciLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopulateFtvacctLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopulateFtvactvLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopulateFtvatypLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopulateFtvcoasLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopulateFtvctypLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopulateFtvfundLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopulateFtvlocnLovd()`	Populates the dynamic/run time version of the Record Group.

Name	Function
`gPopulateFtvorgnLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopulateFtvprogLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopulateFtvprojLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopulateFtvruclLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopulateGxrdirdLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopulateGxrxvbank()`	Populates the dynamic/run time version of the Record Group.
`gPopulateRoiaisyLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopulateTbbdetcLovd()`	Populates the dynamic/run time version of the Record Group.
`gPopUpMenu()`	Populates and clears popup menus.
`gQueryOnlyRole()`	Determines if the current page is running in query-only mode.
`gQuickflow()`	Launches and executes a QuickFlow.
`gReadMetadata()`	Retrieves metadata for the current page.
`gReconnect()`	Reestablishes database connection if possible.
`gResetGlobal()`	Resets the global variables for pop-up windows.
`gResetView()`	Resets the position for pop-up windows.
`gResizeWebWindow()`	Resizes browser windows that are too small.
`gResynchRecord()`	Calculates and resynchronizes a block’s activity date to prevent date-related errors with APIs.
`gSearch()`	Package used with the new code/description search mechanism.
`gSearchWhere()`	Package used with the new code/description search mechanism.
`gSecuredFormCall()`	Performs secured page calls.
`gSecuredFormCallPl()`	Performs secured page calls.
`gSelSobseqnMaxseqno()`	Returns the current sequence number in table SOQSEQN for the `sobseqn_function` argument.

Name	Function
`gSelSpridenId()`	Returns the current ID for the PIDM argument passed when invoked.
`gSelSpridenIdName()`	Replaced by `gValidId();`.
`gSelSpridenPidmName()`	Replaced by `gValidId();`.
`gSetItem()`	Disables or enables an item when passed a valid item name.
`gSetMenu()`	Disables or enables an menu item.
`gSetInstProperty()`	Displays the instance name in window title bar.
`gSetUserPreferences()`	Stores user preference values.
`gSetWinProperty()`	This procedure is now null.
`gShowMenu()`	This procedure is now null.
`gShowMenuBkshlf()`	Displays bookshelf when called from menu item.
`gStartup()`	Start-up trigger for validation page.
`gTimerExp()`	Handles the options timer and the bubble help timer.
`gToolbar()`	Executes the appropriate task associated with the setting of the toolbar.
`gTracePkg()`	Routine which is used for debugging purposes.
`gUpdateActivityDate()`	Updates the activity date column in the table associated with the current block.
`gValidateFixedLength()`	Checks the length of fixed-length data fields.
`gValidAllId()`	Validates person and non-person, and checks for `deceased` and `confidential` flags.
`gValidId()`	Validates person.
`gVerifyIdExists()`	Checks for the existence of a specific ID.
`gVpdi()`	The main package supporting Virtual Private Database Indicator (VPDI) processing.
`gVpdiTrigger()`	Executes baseline VPD procedures.
`gWalkForm()`	Walks through all items in all blocks of a page.
`gWebShowDocument()`	Creates a web document and displays it in a separate browser window.
`gWinActivated()`	Executes `gSetInstProperty();`.
`gWinClosed()`	Closes the event window.
`gWinDeactivated()`	This procedure is now null.
`gWriteBlock()`	Writes the records from a block to a flat file.

GOQCLIB

To maintain consistency, Banner's identification pages all reference a Banner library called the Common Forms Object Library (GOQCLIB). This library is a page object, but not accessed directly by users. The `general/common/libraries/Goqclib` stores common page elements that display on the General Person Identification (SPAIDEN) page and many other pages.

See "Basic Person", of the *Banner General User Guide* for details about the common elements found in `general/common/libraries/Goqclib`.

Workflow Banner Adapter Library (GOQWFLW)

Banner pages are delivered with a live library named `general/common/libraries/Goqwflw`, the single repository for all baseline, cross-product Banner Workflow functionality available within Banner when it is invoked from Banner Workflow.

To guarantee that the required Workflow functionality can be accessed within each page, this library is attached to every baseline Banner page that is defined as a component to Banner Workflow.

Oracle Advanced Queuing

Oracle Advanced Queuing (AQ) is Oracle's message broker implementation that supports asynchronous messaging. AQ is the preferred technology supporting application integration, because it provides database-integrated message queuing.

AQ provides a store and forward capability that guarantees the successful delivery of messages to interested applications. The applications do not need to be running when a Banner Event is created to receive the message triggered by the Event.

Banner uses Oracle AQ because it has the flexibility to support any asynchronous communication with systems external to Banner and is a feature that is included with Oracle Enterprise Edition. AQ eliminates the need for a proprietary message broker.

The Banner Event architecture uses Oracle AQ to store Banner Event messages. These are XML messages that describe an event that has occurred in Banner. Interested applications may consume Banner Event messages and act on them.

The Banner Entity API (package `gb_event`) generates event messages when an entity is created, updated, and deleted. In the future, other Banner APIs may also generate Banner Events to indicate that a specific business process has occurred.

AQ is required for LDI (Luminis Data Integration) version 1.1 for e-Procurement, and also for OpenEAI-based integrations. In the future, other applications may require AQ functionality for integrations with Banner.

AQ support is provided by Banner General 6.2.2, an optional release which supports LDI (Luminis Data Integration) version 1.1 for e-Procurement. Release 7.0 introduces the Banner General objects that support Banner Events, and Release 7.1 includes modifications to several of the general objects to support LDI version 1.1 for e-Procurement. Releases 7.0 and 7.1 require Oracle 9.2.0.4. Oracle 9.2.0.5 is required to fully implement Banner Events.

Clients must configure Oracle AQ only if using LDI for e-Procurement version 1.1 messaging integration or any future Banner certified messaging integration.

Note: The `gb_event` package does not contain a hard-coded reference to the Oracle AQ queue names, so this package will compile without errors if AQ is not configured. However, if events are enabled system-wide on the GUAINST page (`gubinst.gubinst_message_enabled_ind`) and enabled for the specific event through the GURMESG page, and Oracle AQ is not configured, a run-time error will occur when attempting to store a Banner Event XML message to Oracle AQ.

If you are setting up Oracle AQ, it requires a separate tablespace, which must be named `BANAQ`. This is because of a documented restriction on AQ under Oracle 9i. An Oracle persistent queue's data is stored in a table that is mapped to a tablespace. The tablespace used to store the Oracle table will not be able to provide tablespace point-in-time recovery. The `gb_advq_util` package will expect a tablespace named `BANAQ` when creating the queues and queue tables.

For more information on setting up Oracle AQ for LDI for e-Procurement, see *Banner AQ Bridge for LDI for e-Procurement 1.1 Installation and Configuration Guide* and *Banner AQ Connection for LDI for e-Procurement 1.1 Configuration Guide*.

Considerations for building custom applications

You must consider the following factors for building custom applications.

Storing internal LOBs

To use internal LOBs, you must create a separate tablespace for temporary LOB processing. It is recommended that you make your temporary LOB tablespaces extendable.

The General 7.3 installation included creating a Large Object tablespace named BANLOB, with Autoextend `On` by default. If you do not expect to load large objects into the database at this time, you can change the default 1000M allocation for this tablespace to a smaller number.

Storing BFILEs

While we support both internal LOBs and BFILEs, Ellucian strongly recommends that you store your data as internal LOBs. (For the eBill enhancement, you make this decision when you store the Statement files.)

For Banner SaaS deployments consult with your cloud administrator for best practices and recommendations when using the BFILE functionality.

There are many issues with storing files in the file system that should be understood before choosing the BFILE option.

Choosing between internal LOBs and BFILEs

Large object data can consume large amounts of disk space. The disk space will be used regardless of whether the LOB data is managed inside the database or outside on the file system.

However, internal LOB storage provides many features such as:

- `Transaction recovery`--LOB data is committed and rolled back like any other data.
- `Backup`--LOB data is backed up and restored like any other data in the database.
- `Security`--Security is provided by the baseline Banner security rather than server file system security.
- `Space management`--The LOB data can be managed in a separate tablespace specifically allocated for this purpose.

BFILE storage is an option for clients who require the data to be stored outside the database, but who need to access it from within the database. The interface provided by the `gb_large_object` package makes the physical storage location of the data transparent to the application program.

If you choose to use BFILE storage, carefully consider these issues:

- FILE locators do not participate in database exports. In other words, only the locator is exported, not the data.
- The data in the file system is read-only, so BFILEs cannot be updated.
- When an application purges index data from its local table (i.e. TBBSTMT), rather than delete the corresponding GORBLOB row, it will flag that row for later deletion if the data is stored as BFILE. The file system file is *not* deleted. Only the GORBLOB record is flagged as deleted. The purging of the file system files is a system administration task. The system administrator will

then have to use a script similar to the following script to identify and delete the file system file, and then go back and delete the GORBLOB rows.

```
/*This is a sample script you might use if you are storing large
objects as BFILES, and the application has flagged the GORBLOB record
for deletion.

After you run this and obtain the list of files no longer being
referenced by the application, remove them from the file system, and
then delete the gorblob rows.*/

set serveroutput onDECLARE

lv_file_name VARCHAR2(100);

BEGIN

FOR purged_bfiles in (

SELECT gorblob_media_id

FROM gorblob

WHERE gorblob_bfile IS NOT NULL

AND gorblob_deleted = 'Y') LOOP

lv_file_name:=
gb_large_object.f_get_bfile_location(purged_bfiles.gorblob_media_id);

dbms_output.put_line( 'rm '||lv_file_name);

END LOOP;

end;

/
```

Banner Integration

This chapter discusses the objects outside the General product that are shared with the other Banner products.

Common tables

The following is a list of all common tables that are shared by all products within Banner.

Table	Description
PTRTENR	Faculty Member Tenure Status Code Table
SHBCOMI	Committee Information Table
SHBCRMY	Ceremony Information Table
SHRCOMC	Committee Comments Table
SHRCOMM	Committee Information Table
SIBCFTE	Faculty Work load Contract FTE Rule Table
SIBFACD	Faculty Information Table
SLBBLDG	Location/Building Description Table
SLBEVNT	Event Base Table
SLBRDEF	Room Description Table
SLRBCAT	Room Category Definition Table
SLRBCMT	Building Comments Table
SLRBDEF	Building Attributes Definition Table
SLRCMNT	Building/Room Comments Table
SLRCOLC	Room Attributes Collector Table
SLRECMT	Events Comments Table
SLRRASG	Room Assignment Table
SLRRDEF	Room Attributes Definition Table
SLRRUSE	Room Usage Restriction Table
SOBSBGI	Source/Background Institution Base Table
SOBSEQN	Sequence Number Base Table
SORBACD	Source/Background Institution Academic Repeating Table

Table	Description
SORBCHR	Source/Background Institution Characteristics Repeating Table
SORBCMT	Source/Background Institution Comments Repeating Table
SORBCNT	Source/Background Institution Contact Person Repeating Table
SORBDEG	Source/Background Institution Degrees Offered Repeating Table
SORBDMO	Source/Background Institution Demographics Repeating Table
SORBDPL	Source/Background Institution Diplomas Offered Repeating Table
SORBETH	Source/Background Institution Ethnic Make-up Repeating Table
SORBTST	Source/Background Institution Test Score Repeating Table
SORCONC	Prior College Concentration Area Repeating Table
SORDEGR	Prior College Degree Table
SORFADR	Fin. Aid Data Reconciliation Table
SORGEOR	Geographic Region Rules Table
SORMAJR	Prior College Major Repeating Table
SORMINR	Prior College Minor Repeating Table
SORPCOL	Prior College Table
SPBPERS	Basic Person Base Table
SPRADDR	Address Repeating Table
SPRCOLR	Person Collector Table
SPREMRG	Emergency Contact Repeating Table
SPRHOLD	Person Related Holds Repeating Table
SPRIDEN	Person Identification/Name Repeating Table
SPRTELE	Telephone Table
SSBSECT	Section General Information Base Table
SSRMEET	Section Meeting Times Repeating Table
SSRXLST	Cross List Section Repeating Table
STVACAT	Award Category Validation Table

Table	Description
STVACCG	Activity Category Validation Table
STVACTC	Student Activity Validation Table
STVACTP	Activity Type Validation Table
STVACYR	Academic Year Validation Table
STVADMR	Admission Request Code Validation Table
STVASCD	Room Assignment Status Code Validation Table
STVASRC	Address Source Code Validation Table
STVASTY	Assignment Type Validation Table
STVATYP	Address Type Validation Table
STVBCHR	Background Institution Characteristics Validation Table
STVBLDG	Building Code Validation Table
STVCAMP	Campus Validation Table
STVCIPC	CIP Code Validation Table
STVCITZ	Citizen Type Validation Table
STVCNTY	County Code Validation Table
STVCOLL	College Validation Table
STVCOMF	Committee Function Code Table
STVCOMS	Committee Status Code Table
STVCOMT	Committee Type Code Table
STVDAYS	Day of Week Validation Table
STVDEGC	Degree Code Validation Table
STVDEPT	Department Validation Table
STVDISA	Disability Type Validation Table
STVDLEV	Faculty Member Degree Level Validation Table
STVDPLM	Diploma Type Validation Table
STVEMPT	Employment Type Validation Table
STVETCT	IPEDS Ethnic Validation Table
STVETHN	Ethnic Code Validation Table
STVETYP	Event Type Validation Table
STVFCNT	Faculty Contract Type Validation Table
STVGEOD	Geographic Region Division Validation Table

Table	Description
STVGEOR	Geographic Region Validation Table
STVHLDD	Person Hold Type Validation Table
STVHOND	Degree Honors Validation Code Table
STVHONR	Academic History Departmental Honors Validation Table
STVINIT	Recruiting Initials Code Validation Table
STVLANG	Native Language Validation Table
STVLEAD	Leadership Validation Table
STVLEVL	Student Level Validation Table
STVLGCY	Legacy Code Validation Table
STVMAJR	Major, Minor, Concentration Validation Table
STVMATL	Recruiting Material Code Validation Table
STVMDEQ	Medical Equipment Code Validation Table
STVMEDI	Medical Code Validation Table
STVMRTL	Marital Status Validation Table
STVNATN	Nation Validation Table
STVORIG	Originator Validation Table
STVPENT	Port of Entry Validation Table
STVPRCD	Phone Rate Code Validation Table
STVPTYP	Person Type Validation Table
STVRDEF	Building/Room Attributes Validation Table
STVRELG	Religion Code Validation Table
STVRELT	Relationship Validation Table
STVRMST	Room Status Code Validation Table
STVRRCD	Room Rate Code Validation Table
STVSBGI	Source/Background Inst Validation Table
STVSITE	Site Validation Table
STVSPON	International Sponsor Validation Table
STVSPSR	Disability Type Validation Table
STVSTAT	State Code Validation Table
STVSUBJ	Subject Validation Table
STVTELE	Telephone Type Validation Table

Table	Description
STVTERM	Term Code Validation Table
STVTESC	Test Score Validation Table
STVTRMT	Term Type Validation Table
STVVTYP	Visa Type Code Validation Table

Common objects

The following is a list of common objects shared by all products.

Object Name	Object
aofacon.sql	`f_alumni_constituent_ind` function
aofaorn.sql	`f_alumni_organization_ind` function
aoffrdn.sql	`f_alumni_friend_ind` function
comview.sql	driver script to compile all common views
foffagn.sql	`f_finance_agency_ind` function
foffban.sql	`f_finance_bank_ind` function
foffcun.sql	`f_finance_customer_ind` function
foffden.sql	`f_get_finance_desc` function
foffemn.sql	`f_finance_employee_ind` function
foffmgn.sql	`f_finance_manager_ind` function
fofforn.sql	`f_get_special_finance_desc` function
foffven.sql	`f_finance_vendor_ind` function
fofusrn.sql	`f_finance_user_ind` function
pofhapn.sql	`f_payroll_applicant_ind` function
pofhben.sql	`f_payroll_beneficiary_ind` function
pofhcbn.sql	`f_payroll_cobra_ind` function
pofhemn.sql	`f_payroll_employee_ind` function
pofheon.sql	`f_get_eeoc_description` function
ptrtenr page	Tenure Code Rule page
rofrapn.sql	`f_finaid_applicant_ind` function
rofratn.sql	`f_fa_amt_term_func` function
rofrayn.sql	`f_fa_amt_uear_fun` function

Object Name	Object
rofrcsn.sql	`f_sem_csed_fun` function
rofrden.sql	`f_finaid_get_desc` function
rofrfcn.sql	`f_family_contrib_fnc` function
rofrfin.sql	`f_family_income_fnc` function
rofrian.sql	`f_inst_aid_fnc` function
rofrpcn.sql	`f_parent_contrib_fnc` function
rofrpyn.sql	`f_authorized_payments` function
Shacomi page	Committee/Service page
Shicmbq page	Committee/Service Member Inquiry page
Shicmid page	Committee/Service by Person Inquiry page
Shicomq page	Committee/Service Inquiry page
shvcomi.sql	Committee Query View
shvcomm.sql	Committee Member Query View
Slabldg page	Building Definition page
Slabqry page	Building Query page
Slaevnt page	Event page
Slardef page	Room Definition page
Sliaevn page	Event Available Room Query page
Slqbcat page	Building Category Query page
Slqevnt page	Event Query page
Slqroom page	Room Query page
Soacomp page	Non Person Search page
Soaddrq page	Address Summary page
Soageor page	Geographic Region Rules pages
Soahold page	Hold Information page
Soaiden page	Person Search page
Soaigeo page	Geographic Regions by ID page
Soaqgeo page	Geographic Region Query page
Soasbgi page	Source/Background Institution Base page
sofsadn.sql	`f_student_admissions_ind` function
sofsapn.sql	`f_applied_for_degree` function
sofscdn.sql	`f_get_class_desc` function

Object Name	Object
sofscln.sql	`f_class_calc_fnc` function
sofsden.sql	`f_student_get_desc` function
sofseln.sql	`f_enrolled_this_term` function
sofsern.sql	`f_student_enrollment_ind` function
sofsfan.sql	`f_student_faculty_ind` function
sofsgrn.sql	`f_graduated_from_institution` function
sofsgsn.sql	`f_student_gen_students_ind` function
sofshcn.sql	`f_get_hsch_code` function
sofshin.sql	`f_high_school_rowid` function
sofshon.sql	`f_student_housing_ind` function
sofsrcn.sql	`f_student_recruit_ind` function
sofsren.sql	`f_student_registration_ind` function
sofsrgn.sql	`f_registered_this_term` function
sofstdn.sql	`f_sgbstdn_fields` function
sofstrn.sql	`f_student_transfer_work_ind` function
sofstsn.sql	`f_get_sortest_rowid` function
sofstun.sql	`f_get_sgbstdn_rowid` function
Soisbgi page	Source/Background Institution Query Only page
Soqhold page	Holds Query Only page
Soqmenu page	Student Menu page
sovcolp.sql	Prior College Information View
sovconc.sql	Prior College Concentration Area Information View
sovdegr.sql	Prior College Degree Information View
sovgeor.sql	Geographic Region View
sovmajr.sql	Prior College Major Information View
sovminr.sql	Prior College Minor Information View
sovsbgr.sql	Source/Background Institution Base Information View
spvaddf.sql	Address Hierarchy View for FOCUS
spvaddi.sql	Addresses for BannerQuest View
spvaddr.sql	Address Hierarchy Selection View

Object Name	Object
spvadds.sql	Address Hierarchy View
spvcurr.sql	Current PIDM, ID, and Name Information View
spvintl.sql	Person International Information View
spvmedi.sql	Person Medical Information View
ssamatx page	Building/Room Schedule page
ssvmeet.sql	Section Meeting Time View
stkcomf.sql	Cursor `stvcomfc`
stkcoms.sql	Cursor `stvcomsc`
stkcomt.sql	Cursor `stvcomtc`
stkhond.sql	Cursor `stvhond`
Stvacat page	Degree Award Category Code Validation page
Stvaccg page	Activity Category Validation page
Stvactc page	Activity Code Validation page
Stvactp page	Activity Type Validation page
Stvacyr page	Academic Year Validation page
Stvadmr page	Admission Request Checklist Code Validation page
Stvascd page	Room Assignment Status Code Validation page
Stvasrc page	Address Source Validation page
Stvasty page	Assignment Type Code Validation v
Stvatyp page	Address Type Code Validation page
Stvbchr page	Background Inst. Characteristic Code Validation page
Stvbldg page	Building Code Validation page
Stvcamp page	Campus Code Validation page
Stvcipc page	CIPC Code Validation page
Stvcitz page	Citizen Type Code Validation page
Stvcnty page	County Code Validation page
Stvcoll page	College Code Validation page
Stvcomf page	Committee Member Role/Function Validation page
Stvcoms page	Committee/Service Status Validation page
Stvcomt page	Committee/Service Type Code Validation page

Object Name	Object
Stvdays page	Days of the Week Validation page
Stvdegc page	Degree Code Validation page
Stvdept page	Department Code Validation page
Stvdisa page	Disability Type Code Validation page
Stvdlev page	Degree Level Code Validation page
Stvdplm page	Diploma Type Code Validation page
Stvempt page	Employment Type Validation page
Stvetct page	IPEDS Ethic Code Validation page
Stvethn page	Ethnic Code Validation page
Stvetyp page	Event/Function Type Code Validation page
Stvfcnt page	Faculty Contract Code Validation page
Stvgeod page	Geographic Region Division Code Validation page
Stvgeor page	Geographic Region Code Validation page
Stvhldd page	Hold Type Code Validation page
Stvhond page	Departmental Honors COde Validation page
Stvhonr page	Institutional Honors Code Validation page
Stvinit page	Initials Code Validation page
Stvlang page	Language Code Validation page
Stvlead page	Leadership Validation page
Stvlgcy page	Legacy Code Validation page
Stvmajr page	Major, Minor, Concentration Code Validation page
Stvmatl page	Material Code Validation page
Stvmdeq page	Medical Equipment Code Validation page
Stvmedi page	Medical Code Validation page
Stvmrtl page	Marital Status Code Validation page
Stvnatn page	Nation Code Validation page
Stvorig page	Originator Code Validation page
Stvpent page	Port of Entry Code Validation page
Stvprcd page	Phone Rate Code Validation page
Stvptyp page	Source Contract Person Type Code Validation page

Object Name	Object
Stvrdef page	Building/Room Attribute Code Validation page
Stvrelg page	Religion Code Validation page
Stvrelt page	Relation Code Validation page
Stvrmst page	Room Status Code Validation page
Stvrrcd page	Room Rate Code Validation page
Stvsbgi page	Source/Background Institution Code Validation page
Stvsite page	Site Code Validation page
Stvspon page	International Student Sponsor Code Validation page
Stvspsr page	Disability Service Code Validation page
Stvstat page	State/Province Code Validation page
Stvsubj page	Subject Code Validation page
Stvtele page	Telephone Type Validation Code page
Stvterm page	Term Code Validation page
Stvtesc page	Test Code Validation page
Stvtrmt page	Term Type Validation page
Stvvtyp page	Visa Type Code Validation page
toftadn.sql	`f_amount_due` function
toftbln.sql	`f_account_balance` function
toftccn.sql	`f_calc_and_call_fnc` function
toftchn.sql	`f_term_charges` function
toftcon.sql	`f_collection_ind` function
toftcrn.sql	`f_cat_range_fnc` function
toftctn.sql	`f_cat_term_fnc` function
toftdan.sql	`f_calc_aged_days` function
toftden.sql	`f_get_ar_desc` function
toftdon.sql	`f_ar_deposit_ind` function
toftdpn.sql	`f_deposit_balance` function
toftdtn.sql	`f_ar_detail_ind` function
toftefn.sql	`f_oldest_effective_date` function
toftfan.sql	`f_financial_aid_memos` function

Object Name	Object
tofthrn.sql	`f_other_range_fnc` function
tofthtn.sql	`f_other_term_fnc` function
toftmen.sql	`f_memo_balance` function
toftmmn.sql	`f_ar_memo_ind` function
toftomn.sql	`f_opt_term_fnc` function
toftorn.sql	`f_opt_range_fnc` function
toftotn.sql	`f_balance_other_terms` function
toftown.sql	`f_amount_owned` function
toftpan.sql	`f_term_payments` function
toftpfn.sql	`f_ar_profile_ind` function
toftrrn.sql	`f_req_range_fnc` function
toftrtn.sql	`f_req_term_fnc` function
toftsln.sql	`f_calc_aging_slot` function

Ethnicity codes in Banner

This section gives you a guide for building and maintaining the tables that store ethnicity data. You should consider these factors when preparing ethnicity data entry for EEO reporting within the Human Resources system and for IPEDS reporting within the Student system.

Ethnic distinctions

The Ethnic Codes Rule (PTRETHN) page and IPEDS Ethnic Validation Table (STVETCT) store information about the ethnic background of individuals.

Note: Institutions can use the IPEDS Ethnic Validation Table to record Federal Government reporting codes. Values not used for official reporting should not be added to STVETCT.

If you need to store more distinctive, perhaps institution–specific, ethnicity descriptions, use the Ethnic Code Validation Table (STVETHN). This table allows you to make further ethnicity distinctions, such as entering Apache, Blackfoot, and Sioux as types of Native American. These lower value codes are then crosswalked into the Human Resources and Student systems against the PTRETHN and the STVETCT pages respectively. This crosswalk mapping ensures proper Federal ethnic values.

Note: Be sure to coordinate the process of maintaining the Ethnic Code Validation Table (STVETHN) between the Student and Human Resources systems, so that you use agreed upon values where appropriate. after you enter values, under no circumstances should you change or delete them to coincide with reports.

Race and ethnicity categories

The U.S. 2000 Census was collected using new race and ethnicity categories, and the EEOC has mandated that Affirmation Action reports for 2005 use this census data for comparison purposes.

Not all U.S. government departments have adopted this requirement. We anticipate that the National Center for Education Statistics (NCES) will eventually release new IPEDS reporting parameters that require institutions to provide information based on the new OMB categories. Thus, institutions should begin the process of collecting the information based on the new categories.

Banner is being updated to collect data based on the new race and ethnicity categories. In addition to the new categories, a person now has the ability to select one or more of the race categories. Currently, Banner only allows one ethnicity per person record on SPBPERS.

Banner's current **Ethnicity Code** will continue to be maintained by Banner on the appropriate person pages. In order to comply with the EEOC, we will release new rules and Human Resources pages to comply with the data collection requirements.

Race code pages

The Regulatory Race Validation (GTVRRAC) table stores regulatory race codes. U.S. government codes were delivered as system required seed data. Use the Regulatory Race Validation (GTVRRAC) page to maintain this table.

Note: These new codes will not be used for the 2005 IPEDS reporting cycle. However, they must be mapped to race codes (see below) for future regulatory reporting.

Institution-defined race codes can be established on the Race Rules (GORRACE) page and are stored on the Race Rules (GORRACE) table. When creating these codes, there should be at least one race code for each of the U.S. government-established regulatory race codes (as mentioned above).

For more information on the pages and tables for the race codes, refer to the *Banner General User Guide*.

The race and ethnicity fields appear on the Biographical window.

You can find more information on how to use the ethnicity and race codes in the Banner Human Resources system.

Nonresident aliens

When dealing with individuals who are nonresident aliens, it is important to be aware of the methods for reporting them in the Student and Human Resources systems.

Student system

The Student System’s IPEDS report will not consider an individual’s ethnic code if the person is a nonresident alien.

An individual achieves nonresident alien status in the Student system if the current visa type established on the International Information (GOAINTL) page for that person has been set up on the Visa Type Code Validation (STVVTYP) page with the Non-Res(ident Alien Indicator) check box selected.

Human Resources system

The Human Resources system will not report an individual’s ethnic code if that person is a nonresident alien.

An individual achieves nonresident alien status in the Human Resources system if both of the following are true at the same time:

- the Citizen code is entered on the Identification (PPAIDEN) page with a corresponding entry in the Citizen Type Validation Table (STVCITZ) and the citizen indicator `STVCITZ_CITIZEN_IND` set to `N`.
- the person exists in the Person International Information Table (GOBINTL) and the Alien Registration Number field `GOBINTL_ALIEN_REG_NUMBER` is null (has no value).

The Human Resources system will report a person’s ethnic code if all of the above hold true except the person’s Alien Registration Number is not null (has a value).

Reports and Processes

This information describes Banner General batch processing (reports, processes and processing attributes), Sleep/wake processing methods, online batch requests, and other miscellaneous information.

Reports in Banner General

Banner General Pro*C, Pro*COBOL and Java reports are listed in "Reports and Processes", of the *Banner General User Guide*.

Perl Reports

Banner General contains the following Perl reports and processes.

`gebcmplc.pl`	General Master Pro*C compile script
`gencmpl.pl`	General Master COBOL compile script
`gjajobs.pl`	Main Job Submission script invoked by the gurjobs C program
`gjajsub.pl`	Called by gjajobs.pl to do the actual submission of a process to the operating system
`gjawnte.pl`	Obtains the Banner and Oracle Windows NT Environment Variables from the NT Registry
`gjawnts.pl`	Spawns gjajsub in the background (Windows NT submit)
`gjpicur.pl`	Text Manager Extract
`gjpjlis.pl`	Purge Saved Report Output from JobSub
`gjrjflu.pl`	Import uploaded file to JobSub Server
`gjrjlis.pl`	Save Job Report Output in GJRJLIS table: optionally convert to PDF. View and download on page.
`gurjsdn.pl`	Import Oracle Directory File to JobSub Server
`gusmdid.pl`	SDE Inquiry and Delete
`glbdata.pl`	Executes GLBDATA
`glblsel.pl`	Executes GLBLSEL
`gjbparm.pl`	Executes GLBPARM and GLOLETT
`glolett.pl`	Executes GLOLETT

`glrletr.pl`	Executes GLRLETR and GUAPRPF
`guavrfy.pl`	Executes GUAVRFY
`gurjobs.pl`	Executes GURJOBS
`gurjwnt.pl`	Opens a background task in Windows and runs gurjobs.pl
`gurplb1.pl`	Takes in a Script Name and a number of seconds to sleep, then calls gurplb2.pl in the background
`gurplb2.pl`	Runs the script passed from gurplb1.pl looping and sleeping at the interval specified, which is also passed from gurplb1.pl
`sctproc.pl`	Banner C compiler
`sctprocb.pl`	Banner COBOL Compiler

Report and Process Attributes

Report and Process Attributes Legend	
Report or Process	The report/batch process name.
Language	Identifies the language for the process - COBOL, C, Java, SQL, or PL/SQL.
Update/Query	Does the process update any tables, or is it strictly a query-only report?
Audit	Can you run the update process in Audit Mode, so that you can produce the report without an update taking place (Yes or No)? *Yes* appears in this column only if the process permits both update and audit mode. If the report is query only, *Yes* does not appear in this column.
Job Submission	Can you run the process through job submission (Yes or No)?
Sleep/Wake	Is the process used in conjunction with Sleep/ Wake (Yes or No)?
Off Peak	Is it recommended that you defer this program to an off-peak processing time (late night, weekends) for performance reasons (Yes or No)?

Report and Process Attributes Legend	
Restart	If the process aborts or is terminated after the process is initiated, are special procedures required to restart the process without any adverse consequences (Yes or No)? *Yes does not appear in this column if the job can be restarted without special procedures.* If *Yes* appears, refer to the Restart section of this chapter for more information regarding recovery procedures.

Report or Process	**Language**	**Update/ Query**	**Audit**	**Job Submission**	**Sleep/ Wake**	**Debug/ Trace**	**Off Peak**	**Restart**	**Mode**
GJRJFLU	Java	Update		Yes					
GJPICUR	Java	Update		Yes					1 (extract) vers 2 (apply)
GJPJLIS	Java	Update	Yes	Yes					
GJRJLIS	Java	Update		Yes					
GJRRPTS	C	Query		Yes				Yes	
GLBDATA	COBOL	Update				Yes		Yes	
GLBLSEL	COBOL	Update				Yes	Yes	Yes	
GLBPARM	COBOL	Query		Yes				Yes	
GLOLETT	COBOL	Update		Yes		Yes		Yes	
GLRLETR	C	Update	Yes	Yes			Yes	Yes	
GPPADDR	C	Update		Yes				Yes	
GORPGEO	C	Update	Yes	Yes	Yes				
GORSEVE	C		Yes	Yes					
GORSGEO	C	Update	Yes	Yes	Yes				
GUAVRFY	COBOL	Query		Yes				Yes	
GUAGETP	COBOL	Query						Yes	
GUASETR	COBOL	Query						Yes	
GUPDELT	C	Update	Yes	Yes		Yes			
GURDETL	C	Update		Yes		Yes			
GURHELP	C	Query		Yes				Yes	
GURINSO	C	Update						Yes	

Report or Process	Language	Update/ Query	Audit	Job Submission	Sleep/ Wake	Debug/ Trace	Off Peak	Restart	Mode
GURJSDN	Java	Update		Yes					
GURPDED	C	Query		Yes				Yes	
GURTABL	C	Query		Yes				Yes	
GURTEXT	C	Query		Yes		Yes			
GURTPAC	C	Update	Yes						
GUSMDID	C	Update	Yes	Yes					

Trace mode (debug) for General COBOL programs

Running a process in trace mode provides you with a step-by-step process history. It can be used to track down the source of an error message or to verify your place in the process.

Note: Trace mode is not available when you use Job Submission (see restrictions in the Report and Process Attributes Matrix) to run the process.

The following General COBOL programs may be executed in trace mode:

GLBLSEL - Letter Generation Variable Data Extract Process	
Three options are available for trace mode. These options are controlled by the value of the debug flag parameter which is passed on the command line.	
UNIX:	glblsel.shl userid password 1 GLBLSEL Y glblsel.shl userid password 1 GLBLSEL I glblsel.shl userid password 1 GLBLSEL S
Debug flag:	`Y` – display SQL, paragraph names and additional information `I` – display SQL and values inserted into the GLRCOLR table `S` – display SQL only
Windows	perl -S glblsel.pl userid password 1 GLBLSEL Y or cd %BANNER_HOME%\general\misc perl glblsel.pl userid password 1 GLBLSEL Y **Note:** The -S tells perl to look for the glblsel.pl in the PERL5LIB directory.

GLBDATA - Population Selection Extract Process	
UNIX:	glbdata.shl userid password 1 GLBDATA Y
Windows:	perl glbdata.pl userid password 1 GLBDATA Y

GLOLETT - Automatic Letter Compilation Process	
UNIX:	glolett.shl userid password 1 GLOLETT DEBUG
Windows:	perl glolett.pl userid password 1 GLOLETT DEBUG

For UNIX: In each of the above examples, | tee outputfilename are optional arguments that may be passed at the end of the command line examples. Adding | tee outputfilename will result in the output displayed on the screen to be simultaneously written to a file with the designated outputfilename. This file may be edited and searched for specific messages and errors.

For Windows: The Windows equivalent of the tee command is available with the purchase of the `Windows Services for UNIX Add-On Pack`. Please see http://www.microsoft.com/technet.

SQL*Plus scripts

The following General SQL*Plus procedures are provided to assist you.

`delrslt.sql`	Delete rows from GJBRSLT table.
`dyndflt.sql`	Default parameters for dynamic SQL procedures.
`gchkbgrt.sql`	Builds grants for the security owner to give it full access to all Banner tables installed for which it has no grants at all.
`gchkemail.sql`	Validates e-mail addresses on file in GOREMAL to ensure that an e-mail address will have an ampersand (@) and a period (.). The script also lists duplicate e-mail addresses that are found based upon case insensitivity per PIDM per e-mail type at the end of the report.
`gchksec.sql`	This SQL routine tests for all requirements for role-level security.
`gchksecrole.sql`	A SQL routine to test for local roles that do not adhere to the Banner suggested naming conventions that are used in providing access to Banner pages.
`gchksyn.sql`	Generates an SQL routine to create all missing Banner public synonyms.

gchkuser.sql	A script called by GUPUSER to verify that the upgrade_owner has all the required database objects.
gcreuser.sql	Creates the upgrade_owner and its required database objects.
gdeleqer.sql	Deletes rows from the Event Queue Error table based on a date.
gdeleqrc.sql	Deletes rows from the Event Queue Transaction tables based on a date.
gdelintl.sql	Implementation of Multivisa 5.5. Deletes rows from the international tables GOBINTL, GORVISA and GORDOCM from SPRINTL. Revised for 7.3 redesign. Run before gselvisa.sql, gupdvisa.sql, and gdelsdaxvisa.sql.
gdeljobs.sql	Removes rows from job submission tables for products not available on your system.
gdeloutp.sql	Deletes rows from the Jobsub Database Output tables based on a date.
gdelprun.sql	Deletes leftover GJBPRIN entries for the GLOLETT program before recompiling. Used during upgrades only.
gdelsdaxvisa.sql	Deletes GTVSDAX international rows. Run after gselvisa.sql and gupdvisa.sql.
gdiscon.sql	Disables constraints that should remain disabled.
gdroptab.sql	Drops the GUBSMOD and GURSSQL tables before importing them. Used only during upgrades.
gdrpsyn.sql	Drops public synonyms for Banner objects which no longer exist. Used only during upgrades.
gefixadd.sql	SQL*Plus script to set the from_date on SPRADDR records to the activity date when both from and to dates are null. Only to be run when BannerQuest is installed.
genalug.sql	Grants option for advancement tables.
gencimg.sql	Grants for courts tables and views.
genfimg.sql	Grants option for finance tables.
genford.sql	General foreign grant driver script.
genforg.sql	Script that contains grants for INTEGMGR (Integration Manager).

genpayg.sql	Grants option for human resources tables.
genresg.sql	Grants option for financial aid tables. Replaces GENFAIG.SQL.
genstug.sql	Grants option for student tables.
gentrag.sql	Grants option for accounts receivable.
gfgacdroppol.sql	Script that drops policies on a table for package GOKFGAC.
gfpiiaddpol.sql	Script that adds policies for tables identified in GORFDPI.
gfvbsaddpol.sql	Script that adds FGAC policies for tables identified in GORFDPL.
ggivedba.sql	Script used during the upgrade to alter the users involved with the upgrade to include the DBA role as one of their default roles. The script also generates a file used to restore the roles to what they were before the upgrade.
ggrtfnc.sql	Script can be used to generate grant execute statements to functions for users requiring execute privileges for SDA views.
ggrttmp.sql	Create temporary table used to build grants. This table only exists for the duration of this process.
gindex.sql	Creates a report of indexes for a schema owner.
ginsprun.sql	Inserts gjbprun rows for variables to be recompiled. Used only during upgrades.
gletgrts.sql	Upgrade script to give the General product the ability to run GLOLETT and GLBPARM during the upgrade.
glramod.sql	Process to check if specific modifications have been applied to the database.
gmakalt1.sql	GOSTAGE script which invokes guraltg.sql.
gmakgrt.sql	This process builds grants for the table names loaded into the GUBGRNT table.
gmakgrtv.sql	New routine to save grants, then re-issue them from BANINST1.
gnestedv.sql	Utility script that lists nested variables. Used during upgrades.
gnewgrt.sql	Generates end user grants.

`gostage.sql`	This process is the heart of the upgrade process. It determines what modifications in the GUBSMOD and GURSSQL tables have to be applied to your system.
`greadme.doc`	A text file that lists and describes all the scripts in the general/plus directory.
`gresroled.sql`	This script starts a spooled script, gresrole.sql, that was generated by ggivedba.sql to restore the original roles.
`grunsiz.sql`	This process runs all the standard table sizing model scripts. The start for this file is automatically generated by the glramod routine. Used during the upgrade process.
`gsafobj.sql`	SQL routine used during upgrades to register changes to an object if it exists.
`gsanobj.sql`	Adds new objects to bansecr's security tables. Used during upgrades.
`gsaoobj.sql`	Deletes obsolete objects from bansecr's security tables. Used during upgrades.
`gsdrslt.sql`	Deletes any leftover GJBRSLT records for the staging job.
`gselappl.sql`	This script will extract the application and the creator of components of the application so the user will know what to respond with and who to sign on as when running GLBPARM and GLOLETT.
`gselsevs.sql`	Data fields `GORSEVS_ISSUE_COMMENT` and `GORSEVS_TRANSFER_COMMENT` are no longer used for SEVIS reporting. This script allows users to retrieve data from the latest history record.
`gselvisa.sql`	Selects page GORVISA for GTVSDAX records that are retired from use. Run before gupdvisa.sql and gdelsdaxvisa.sql.
`gsirslt.sql`	Inserts a record in the results table to indicate the task completed successfully.
`gskipgrt.sql`	Script which skips the generation of end user grants. Used during the upgrade process.
`gstrslt.sql`	Test if a hosted SQL*Plus routine succeeded. This routine tests if the hosted routine was able to insert a row into the GJBRSLT table. If the row is not found an SQL error is caused that will stop the current routine.

`guidmod.sql`	Insert history record into the GENERAL.GURDMOD table. Used during upgrades.
`guitmod.sql`	Insert history record into the GENERAL.GURDMOD table if the mod has not already been recorded. Used during upgrades.
`guovmods.sql`	Script to create view GUVMODS under the upgrade_owner created through gupuser.sql.
`gupdintl.sql`	Conversion script to migrate data from SPRINTL in General 5.5. Conversion script revised for the redesign of GORVISA for 7.3.
`gupdvisa.sql`	Script to updated gorvisa and gordocm columns to null for old GTVSDAX international values. Run after gselvisa and before gdelsdaxvisa.sql.
`gupuser.sql`	Script to create an upgrade user account to be used in parallel upgrades.
`guraltb.sql`	This utility script will spool off a sqlplus script to compile all not valid functions and views owned by BANINST1.
`guraltg.sql`	This utility script will spool off a sqlplus script to compile all not valid functions and views. Used by the gostage process.
`guraltr.sql`	This utility script will spool off a sqlplus script to compile all not valid functions and views.
`guramod.sql`	Create the table used to build the modification scripts.
`gurcmnt.sql`	Creates COMMENT ON COLUMN statements for a table in proper format to an Oracle directory.
`gurcmod.sql`	Process to check if specific modifications have been applied to the database. Information is placed into the guramod table indicating what scripts have been executed already and what ones still have to be run.
`gurcmpa.sql`	Spools a script to compile all database objects that are not owned by either SYS or SYSTEM.
`gurconsumer.sql`	Script to provide privileges to enqueue and dequeue messages to Oracle users.
`gurcrypt.sql`	Script to encrypt all passwords.
`gurddoc.sql`	Script to extract database object comments.
`gurdlid.sql`	Script used to delete all information about a PIDM in the database.

gurdmod.sql	Create the table to track database modifications.
guremod.sql	This process is executed at the end of each products xREVTAB and xREVIEW script. This process extract information stored in the guramod table for this user ID and then executes it.
gurespl.sql	Script to generate an exit statement and close the spool file.
gurethnicity.sql	Script to capture race and ethnicity in SPBPERS and GORPRAC.
gurfgrt.sql	Generates an intermediate sql routine that will issue a foreign grant for a table only if it exists. You must be logged on as system to use this routine.
gurgfix.sql	Script to generate and create any missing grants after the security patch has been applied.
gurgfix2.sql	Script to generate and create any missing grants after the security patch has been applied. Use this script instead of GURGFIX if your institution does not use Banner Self-Service.
gurgrnt.sql	Creates a file of GRANT statements based on a model user (replaces GRANTS in the ORATOOLS directory).
gurgrta.sql	Script to grant execute privilege on BANINST1_SS9-owned stored procedure/ package passed as the first argument to BAN_DEFAULT_M role.
gurgrtb.sql	Script to grant execute privilege on the BANINST1-owned stored procedure passed as the first argument to Banner owners and roles.
gurgrte.sql	Script to grant execute on the BANINST1-owned stored procedure passed as the first argument to the e~Print user.
gurgrth.sql	Script to grant execute privilege on the BANINST1-owned stored procedure passed as the first argument to local web server user IDs.
gurgrti.sql	Script to grant execute privilege on the BANINST1-owned stored procedure passed as the first argument to the Integration Manager.
gurgrtn.sql	Script to grant execute privilege on BANINST1 owned stored procedure/package passed as the first argument to NLSUSER.

`gurgrts.sql`	Script to grant execute privilege on the BANINST1-owned stored procedure passed as the first argument to the Banner security owner.
`gurgrtsso.sql`	Script to grant execute privilege on GOKDBMS (in support of DBMS_PIPE) to BANSSO (if BANSSO exists).
`gurgrtw.sql`	Script to grant execute privilege on the Web Tailor-owned stored procedure passed as the first argument to the Banner stored procedure owner, the database roles, and the local web server user IDs.
`gurlsid.sql`	Lists of all tables and columns in which a person exists.
`guromod.sql`	This process is executed at the beginning of each products xREVTAB, xREVIEW and xTABCLN script to delete any old entries left in the GURAMOD table for this user ID.
`gurospl.sql`	Script to set SQL*PLUS options and open a spool file names by parm1. This script is always used by a driver script.
`gurrddl.sql`	Script to PL/SQL script which generates DDL syntax for a specified table(s). This script was made obsolete in Release 8.1 favor of data definition language (DDL) tools provided by Oracle. Oracle's Metadata API and DBMS_METADATA package provide more extensive functionality than gurrddl.sql did, and will remain current with future Oracle updates. For more information, see *Oracle Database Utilities* and *PL/SQL Packages and Types Reference* in Oracle's technical documentation.
`gurrhmu.sql`	Script that invokes a refresh of the hierarchial menu table GURHMNU.
`gursava.sql`	SQL routine that creates SQL*PLUS define commands that contain all the information need to recreate a table the same size and in the same place that it currently exists. Cluster information is not retained.
`gursava2.sql`	SQL routine to save index/table info no matter who owns it.

`gurscls.sql`	This script checks every person enrolled in the class to make sure they have been given execute privileges to every role used by any object in the class.
`gurstop.sql`	Invoke this routine to stop job submission.
`gurtgr1.sql`	This routine is used by the GURTGRT routine to build grants for a new table based on grants for an existing table.
`gurtgr2.sql`	Copy Best Guess generated grants from the work table to a spool file. The output from this select is ordered by grantor to reduce the number of connect commands that must be executed.
`gurtgr3.sql`	Generate report for the best-guess grants generated by GURTGRT and GURTGR1.
`gurtgr4.sql`	Script to generate grants for views using tables as model. Used during upgrades.
`gurtgr5.sql`	Script to generate grants for tables based on another owners table. Used during upgrades.
`gurtgrt.sql`	Generates an intermediate sql routine that will create grants to access a new table based on existing grants for a similarly used table. You have the option of using up to three tables to match. You should be logged on as the grantor (owner of the new table) to run this routine.
`gurtgrto.sql`	Used in conjunction with GURTGR5.
`gurtgrtv.sql`	Used in conjunction with GURTGR4.
`gurtlst.sql`	Produces list and description of a Banner product's tables.
`gurtprt.sql`	Prints contents of a specified table.
`gurutlrp.sql`	This utility script calls Oracle’s utlrp routine which validates database objects in dependency order. A report is spooled.
`gurvlst.sql`	Produces list and description of all validation tables in a Banner product.
`gutemod.sql`	This process is executed at the end of each products xREVTAB script. This process extract information stored in the guramod table for this user ID and then executes it.
`gutfmod.sql`	Process to prime the gurdmod table if the constraint exists.

`gutnmod.sql`	This script will insert a row in the gurdmod if the specified object does not exist. This would be used to conditionally run an upgrade script only if the table is present.
`gutpmod.sql`	Process to prime the GURCMOD table based on an objects existence.
`guttmod.sql`	Process to prime the GURCMOD table based on columns existence.
`iobseqn.sql`	Primes the SOBSEQN table after creation.
`login.sql`	Default SQL*Plus Login parameters.
`repdflt.sql`	Default SQL*Plus parameters to produce a report.
`sleepcms.sql`	This is a generic SQL process for VM/CMS which causes operating-system-dependent command procedures to be executed for batch processes which require sleep/wake capabilities.
`sleepdec.sql`	This is a generic SQL process for OpenVMS which causes operating-system-dependent command procedures to be executed for batch processes which require sleep/wake capabilities.
`sleepunx.sql`	This is a generic SQL process for UNIX which causes operating-system-dependent command procedures to be executed for batch processes which require sleep/wake capabilities.

Sleep/wake methods

Banner provides two different methods for running jobs in a cyclical, or sleep/wake, mode.

Method One

The first method uses OS command scripts and an SQL*Plus script to cause the job to run in a cyclical fashion. These jobs must be submitted from the operating system prompt and must be terminated manually.

To compile programs to run in this fashion, you must define `NO_SLEEP_SW` as a pre-compiler directive to exclude the code used by the second technique.

Seven programs are affected by the value `NO_SLEEP_SW` as a pre-compiler directive:

- `sfrschd.pc`
- `shrtrtc.pc`

- `tgphold.pc`
- `tgrmisc.pc`
- `tgrrcpt.pc`
- `tsrcbil.pc`
- `tsrssum.pc`

Note that `NO_SLEEP_SW` only affects the Student and Accounts Receivable processes.

UNIX

The first command procedure, `sleepunx`, prompts for parameters needed by the second procedure and SQL*Plus script, `sleepunx.shl` and `sleepunx.sql` respectively.

This procedure then starts (or submits) `sleepunx.shl`, which in turn starts sleepunx.sql. The SQL*Plus script `sleepunx.sql` will spool OS-specific commands to run the job into a file, provided there is actually work to do as determined by the parameters previously entered. When the SQL*Plus script exits, `sleepunx.shl` executes the spool file. The parameters needed by the program are contained in a XXXXXXX.dat file which are read through input redirection when the job executes. The second command procedure `sleepunx.shl` then sleeps for the specified interval, awakes, and loops back to start the SQL*Plus script again.

To define `NO_SLEEP_SW` on UNIX, go to `sctproc.mk` and find the lines:

```
# Other C options

CCOPT=
```

Change these lines to:

```
# Other C options

CCOPT=-DNO_SLEEP_SW
```

Windows

Method one is not valid for Windows platforms.

Method Two

The following Banner systems and processes are valid for the Sleep/Wake processing described in this section.

Banner Student

SFRSCHD- Student Schedules

SHRTRTC- Academic Transcript

Banner Accounts Receivable

TGRRCPT- Account Receipt

About this task

TGRMISC- Miscellaneous Receipt

TSRCBIL- Student Billing Statement (Invoices)

TSRSSUM - Student Transaction Summary Report

Procedure

1. Define printer and print command on the Printer Validation (GTVPRNT) page. In the Printer Code field, enter a name to reference each specific printer that may be used for printing output from sleep/wake processing. In the Command field, enter the correct operating system print command as it would normally be entered from the command line prompt, substituting an @ (at sign) as the place holder for the file name to be printed.

   UNIX example: `lp -d talaris1 @`

   OpenVMS example: `print/queue=ln01 @`

   Windows example: `print /d:\\sctrnt0\XeroxDC230 @`
2. On the appropriate System Distribution Initialization Information (SOADEST page for Student or TOADEST for Accounts Receivable), enter the printer Code from GTVPRNT that should be identified with the collector table rows that will be inserted to the appropriate tables when on-line application pages create a request for output that can be generated by sleep/wake processing.

   The collector tables are as follows:

Process	Collector Table
SFRSCHD	SFRCBRQ
SHRTRTC	SHTTRAN
TGRMISC	TBRCMIS
TGRRCPT	TBRCRCP
TSRCBIL	TBRCBRQ
3. On the Process Submission Control (GJAPCTL) page, for the valid sleep/wake jobs listed previously, enter the correct response for the parameter that specifies that the job should be processed for collector table entries. Refer to the documentation for each specific process to determine the appropriate response in each case (correct responses may be COLLECTOR, Y, %, etc.). In addition, each sleep/wake job has a printer code parameter. You must specify exactly the same code for this parameter answer that was entered on either SOADEST or TOADEST. Enter `Y` for the run in sleep/wake mode parameter and specify the number of seconds for the sleep/wake interval (cycle) for each process.

   Note: Do not enter the printer code in the top block of GJAPCTL; only enter it in the parameter section.

4. The Sleep/Wake Maintenance (GJASWPT) page should be used to stop the sleep/wake process or to change the sleep interval. A process name and printer code must be entered in the key. A LIST of values is available in each field to see the valid list of processes and printer codes that have ever been submitted for sleep/wake processing.

   To stop the process, enter `N` in the Continue to Run field and SAVE. The job will not stop immediately, but rather will stop after the next time the process 'wakes up' and finishes the next processing cycle. To change the sleep interval, enter the desired interval in the Next Cycle Time field and save.

   You can also use the GJASWPT page to view statistics regarding how many rows were processed for the most recent wake-up cycle and the total number of rows processed after the process was initiated. You can also determine if the processes terminated abnormally. by viewing the Abnormal Termination field. If there is a `Y` in Abnormal Termination, something caused the process to fail. You should review log files to determine the cause.

Print the saved output

You can enable the Job Submission saved output for Method Two Sleep/Wake processes. This option is only available on non-Windows operating systems. The report files created by the Sleep/Wake process are uploaded to GJAJLIS. The reports may be optionally converted to PDF.

Procedure

1. Enable the Method Two process on the **JobSub Output Definition (GJAJBMO)** page.

   You can adjust the MIME type to PDF and select the appropriate font and font size.
2. Create a printer on the **Printer Validation (GTVPRNT)** page for the print and save the output.
3. If the output is going to be printed externally to the Job Submission server using the Banner Print App, add the printer to the **Local Print Printer Definition (GJALCPR)** page.
4. Set the GTVPRNT command `sh gjrjlis_sw.shl @ printername "lp –p printer name"` for the printer described in step 2 on page 87.

   The @ is a placement for the print output file name. The command parameter printername is the name of the printer written to the GJRJLIS record. The last parameter lp –p printername is the print command executed by the shell `gjrjlis_sw.shl` if the print is to occur from the Job Submission server.
5. Enter the GTVPRNT printer that has the `gjrjlis_sw.shl` command, on the printer destination page.

   The Sleep/Wake process is submitted with same printer in the parameter for the Sleep/Wake printer.

Operating systems without sleep/wake-up commands

Operating systems which do not have sleep commands, or whose sleep commands may not be executed by user programs, must use the Method One.

NOSLEEP Triggers

NOSLEEP Triggers is an alternative method to that of using sleep/wake processing. Placing a trigger on an associated collector table is used to put forth the action of running the desired process on-demand.

Processing jobs through sleep/wake generates a substantial amount of redo log activity. Each time an individual sleep/wake process wakes up to see if there is anything new in the collector table to act upon, an update to a table is performed recording the wake up. Even when there is no activity for the sleep/wake processes to act upon, redo logs continue to fill up and go to disk archival. This is due to the constant wake-up time stamping activity of numerous sleep/wake processes. NOSLEEP Triggers eliminates this excessive redo log/archival log activity, saving significant archive log disk space (in addition to reducing the number of archived logs that would be required for a database restore).

Processing jobs through sleep/wake can also involve starting a process for every printer involved in a particular process. For example, if there are 50 possible receipt printers, there must be 50 sleep/wake processes started to support them. NOSLEEP Triggers eliminates the need to start any such constantly running (cycling) processes.

The implementation of NOSLEEP Triggers is not mandatory nor does its implementation cause a migration to a NOSLEEP Triggers as the only way of processing. As stated, NOSLEEP Triggers is provided as an alternative to sleep/wake. Its implementation can be with as many or as few triggers as required. You can configure some processes to be handled through NOSLEEP Triggers and some other processes to be handled with sleep/wake. The NOSLEEP Trigger method can co-exist with the sleep/wake method. You can switch from a sleep/wake to a NOSLEEP Trigger or vice-versa, for any particular process. However, you need not set up a particular process to run for both sleep/wake and NOSLEEP Triggers processing.

Database packages

Ellucian provides database packages to support NOSLEEP triggers.

GOKNOSL

This package was derived from the Community Source Initiative artifacts (LKH) on February 2010, initial release of primary package in support of the AR segment NOSLEEP Triggers. Package procedures are called from primary AR TOKNOSL package procedures as invoked from corresponding AR NOSLEEP Triggers.

This package simulates the submission of job for processing on behalf of the Oracle user id NOSLEEP. If errors are encountered with gurjobs during trigger processing, they are recorded in gurtklr row (for user id NOSLEEP) and can be viewed using GUAMESG page.

GSPCRPU

This is an added procedure, with logic encapsulated/hidden within the package body, passing in raw and out string in support of NOSLEEP password decryption.

GB_ADVQ_UTIL

Modifications in support of NOSLEEP Triggers Community Source initiative. The syntax PRAGMA AUTONOMOUS_TRANSACTION on procedures `p_enqueue_msg_fragments`, `p_dequeue_msg_fragments`, and `p_dequeue_msg_fragments_condit` was necessary in that these queuing transactransactions are firing within the parent transaction issued from the NOSLEEP Triggers.

Job Submission objects

The following Job Submission related object has been changed.

gjajobs.shl

NOSLEEP Triggers Community Source initiative project. LKH, February 2010. Add sleep delay for NOSLEEP jobs. This is intended to give time for NOSLEEP setups to commit before attempting to retrieve inserted data in GJBPRUN.

Job Submission

While the external mechanics of submitting a job is the same across all operating systems, the internal processing is specific to the operating system. Because of different releases of the OS and local modifications made, these procedures may not run exactly as delivered and may require some modifications.

Before you can submit a job, you must define it on the Process Maintenance (GJAJOBS) page and have the appropriate security privileges granted for the object. Information from this page and from the O/S field found on Installation Control (GUAINST) page control how the command to run the job is built. On the GJAJOBS page, the **Type** field indicates the type of program that executes.

Job Type	Description
C	Pro*C program.
E	Standalone COBOL program. Banner no longer has any type E jobs, the value remains for compatibility. These types of jobs may not use parameters because there is no mechanism provided to pass them.
J	Java batch process. These types of jobs cause operating specific scripts to run that will invoke a batch java process.
P	Procedures. These types of jobs cause operating system specific scripts to run. (Bourne Shell command procedure, or Perl).

Job Type	Description
R	Jasper Report

The Command Name, if entered, will be used as the actual name of the program to run. If it is null, the Name from GJAJOBS will be the name of the program to run. This field should never contain an extension. The extension, if needed, is appended by either the Job Submission Interface (GUQINTF) page or the operating system specific GJAJOBS command procedure.

Jobs may be submitted from either a product's application page or from General's Process Submission Control (GJAPCTL) page.

Note: For more information on processing PL/SQL packages through Job Submission, see Process PL/SQL packages with JOBSUB on page 109.

Jobs submitted from GJAPCTL

The GJAPCTL page provides for the entry and editing of parameters and executes any process level validation associated with the job.

Process level validation is defined on the GJAJOBS page in the **Validation** field and refers to the name of a procedure contained in the product specific validation package stored in the database.

The name of the package is the product's system indicator, as defined on the System Indicator Validation (GTVSYSI) page, appended with the literal `OKPVAL`; General's package is GOKPVAL.

Parameters are inserted into the Process Run Parameter Table (GJBPRUN) using a unique sequence number generated from General's GJBPSEQ sequence to identify the job request. The GJAPCTL page then sets `global.call_form` to GJAPCTL and calls the GUQINTF page (described later).

Jobs submitted from the GJAPCTL page will always use GJAJOBS as the command procedure to run. Depending on your operating system and the type of job running, the GJAJOBS command procedure may further modify the command name passed to it from the GUQINTF page.

For example, in the UNIX environment GJAJOBS.SHL constructs the operating system command as follows:

- For E type jobs - prefixes the jobs name with the COBPREF environment variable and suffixes it with the COBSUFX environment variable.
- For P type jobs - appends the literal, file`.shl` to the end of the command name.
- For C type jobs - does not modify the command name.
- For J type jobs - appends the literal `.shl` to the end of the command name.

If the command name for a type *P* job had specified an extension, another one would be added automatically.

Note: Interactively entering job parameters from the host is no longer supported. Parameters for all jobs must be entered on GJAPCTL.

Reset job submission sequence number

You may reset the job submission sequence number back to a value of 1.

Warning! Before resetting the job sequence number, ensure that a backup is created and the procedure is tested in a TEST system before applying to PROD.

Note: Banner Financial Aid uses GJBPSEQ to generate the log numbers that uniquely identify changes to be processed by the RLRLOGG logging process. Before resetting the job sequence submission number, the Financial Aid “mirror logging tables” must first be emptied by running RLRLOGG to successful completion for all possible aid years. The mirror tables in Banner Financial Aid are as follows:

RLRAPP1

RLLAPP2

RLLAPP3

RLLAPP4

RLLAPP5

To reset the job submission sequence number, execute the following:

```
SQLPLUS GENERAL/PASSWORD
delete from GENERAL.GJBPRUN;
DROP PUBLIC SYNONYM GJBPSEQ;
DROP SEQUENCE GENERAL.GJBPSEQ;
create sequence GENERAL.GJBPSEQ
increment by 1
start with 1
maxvalue 99999999
minvalue 1
nocycle
cache 20
order ;
create public synonym GJBPSEQ for GENERAL.GJBPSEQ;
```

You may also want to clear the GJIREVO database tables related to job submission to avoid inserting duplicate run sequence numbers. To remove all data in the GJIREVO table, execute the following:

```
sqlplus general/password
delete from general.guroutp;
delete from general.guboutp;
commit;
exit;
```

Note: If you are running Appworx, please confirm it is functioning correctly after resetting the sequence.

Jobs submitted from application pages

Requests from application pages typically do not allow you to enter parameters because they are usually obtained from information contained on the page itself. The application page will usually assign values to globals, then call the GUQINTF page. GUQINTF is discussed below.

If the job is to be submitted from a page other than GJAPCTL, a command procedure must exist for the job. If the **Command Name** field on the GJAJOBS page is not used, the job name itself is used as the name of the command procedure. These types of requests are handled by a call to the GUQINTF page which builds the initial host command. Again, depending on the operating system, an extension may be added to the command name before it is executed. Before calling GUQINTF the global named `GLOBAL.JOB_ID` is set to the name (without extension) of the program to be executed and the global named `GLOBAL.CALL_FORM` is set to the name of the current page. If the GJBPRUN table has been populated with parameters, the global named `GLOBAL.ONE_UP_NO`, is set to the one up number used when the rows were inserted.

The GUQINTF page

The GUQINTF page performs several tasks. First, the `GuqintfFormController.java.Guqintf_TaskStarted()` method tests `GLOBAL.CALL_FORM` to see if the request came from GJAPCTL. If so, it executes the `Guqintf_JsHostCommands()` method.

This method builds a message containing the following information:

Command Name	GJAJOBS
`job type`	A one-character code representing the type of job: `E` - executable COBOL program `P` - operating system command procedure `C` - PRO*C `J` - operating system command procedure that launches a batch java process.
`user_name`	Current Oracle username or alternate username if entered on the Alternate Logon Verification (GUAUIPW) page.
`password`	Password for username.
`one_up_no`	One-up number generated from the GJBPSEQ sequence.
`printer name`	Comes from the GJAJOBS page or the GJAPCTL page.
`special pages name`	Comes from the GJAJOBS page or the GJAPCTL page.

Command Name	GJAJOBS
`submit time`	This is from the GJAPCTL page and is currently only passed to the GJAJOBS procedure. No mechanism exists in the procedure to schedule the job due to the wide variety of supported operating systems.

Requests submitted from pages other than GJAPCTL come in two types: those that populate the GJBPRUN table before calling GUQINTF, and those that do not. If the page does not populate the GJBPRUN table, login must exist in the GUQINTF page to do it.

For example, requests coming from the TSASPAY page execute the method `Guqintf_StudentPayment()` which forces an update to the GJBPRUN table.

The block level `gjbprun_AfterRowInsert()` method fires then, executing a common method to actually do the inserts.

The `Guqintf_StudentPayment()` method executes immediately before the `Guqintf_HostCommands()` method in the `Guqintf_TaskStarted()` method. The `Guqintf_HostCommands()` method then builds a message containing the following information:

`command name`	Either the job name or the command name from the GJAJOBS page. Then, based on the operating system as defined on GUAINST, an extension may be added or a prefix may be added. On Windows platforms, perl prefixes the command name. On UNIX, `.shl` is appended.
`user_name`	Current Oracle username or alternate username if entered on the Alternate Logon Verification Form, (GUAUIPW).
`password`	Password for username.
`one_up_no`	One-up number generated from the GJBPSEQ sequence.
`job name`	Either the job name or the command name without an extension (uppercase).
`directory`	Name of the directory where output from the job will go.

After this message is built, the method `Guqintf_Pipeit()` executes which sends the message to the GURJOBS application server program by executing the `DBMS_PIPE.SEND_MESSAGE` function. If the Advanced Queuing alternate communication mechanism has been implemented (an alternative to `DBMS_PIPE`), instead of the `Guqintf_Pipeit()` method being executed, the method `GuqintServices.java.aqit()` is executed, sending the message to the queue `GURJOBS_Q`, which is then dequeue by the GURJOBS application server program.

After sending the message, requests that came from the GJAPCTL page return to that page immediately. Requests coming from other pages perform one more method named `Guqintf_GetStatus()`. This method reads the External Process Results Table (GJBRSLT) to check for a message inserted by the batch job. The lack of an entry results in an error message being displayed stating that the job failed.

Note: If the program does not use the GJBRSLT table, the `Guqintf_GetStatus()` method still needs to be executed because globals are set which indicate success or failure. Returning to the calling page without setting these globals could result in unpredictable results.

When GURJOBS receives the request, it fulfills it by executing the system function, using the command as the argument. See the section on GURJOBS for more information.

The following sections outline the processing for each of the currently supported operating systems.

UNIX

A UNIX shell program called gjajobs.shl is started by the system function.

gjajobs.shl

This shell interrogates the parameters passed to it and builds another temporary shell to actually run the job.

The temporary shell consists of either the commands to execute and print a report, (based on a parameter from the GJAPCTL page), or commands to invoke a customized procedure for this job depending on the definition of the job on the GJAJOBS page.

The `gjajobs.shl` then sets the following environment variables so they can be accessed by the started procedure, if necessary.

`BANUID`	The userid being used to run the job.
`FORM`	Special print options as specified on GJAJOBS or GJAPCTL.
`H`	The HOME directory.
`JOB`	The name of the Job or Process to be executed.
`LOG`	Log file name.
`ONE_UP`	The one up number assigned at the time the job was submitted.
`PRNT`	The name of the Printer as specified on GJAJOBS or GJAPCTL.
`PRNTOPT`	The complete print command built from PRNT and FORM
`PROC`	The name of the .shl file to be run.
`PROG`	The name of the program to run, as indicated in the key block of GJAPCTL.
`PSWD`	The password for the BANUID.

`SUBTIME`	The submit time as specified on GJAPCTL. This parameter is not currently implemented.
`TEMP`	This is the prefix of the generated shl. It is constructed by the concatenation of the process name ($1) and the one up number ($5).
`UIPW`	The concatenation of BANUID/PSWD.

While the variables `PRNT` and `FORM` are made available to the procedure, only primitive print routing and special pages processing are addressed in the shell due to the vast variations in print managers. Customizing will probably be required to conform to the installation specific print programs.

The gjajobs.shl then invokes the generated shell to run the report or customized process as a background process. Control is then returned to the Banner on-line system and the user may continue work while the job executes.

Where possible, the system removes all intermediate and temporary files based on the assumption that jobs run without error in production. The deletion of these files reduces the need for frequent directory maintenance. Occasionally, the need may arise during implementation and training to preserve the intermediate and temporary files to monitor job summary statistics or possible process errors. In this case, you must modify gjajobs.shl so that it does not delete these files.

If you create any new processes of your own, make sure they are accessible through the path of the account used to submit GURJOBS.

umask value for gjajobs.shl

Depending on how your environment is set up, you may want to change the delivered `000` values set by umask for gjajobs.shl to make files more secure.

If jobsub and the users are in different groups you may need to use umask `000`. If they are in the same group you could use `017`. If all reports are run to the database and server access is not required then you could use `077`.

To change the default permissions assigned to your UNIX files and directories, use the umask command. Its format is `umask nnn` where `nnn` is a three-digit code that defines the new default permissions.

Warning! Although they look similar, the umask string does `not` have the same format as the chmod permission string.

Each of the three numbers represents one of three categories: user, group, and other. The value for a category is calculated as follows:

- read (r) permission has a value of 4
- write (w) permission has a value of 2
- execute (x) permission has a value of 1

Sum the permissions you want to set for the category, then subtract that value from 7. As an example, examine the current default umask statement that is used to assign the file protections – `rwx-r-x---` :

`umask 027`

The user category value is `0` because r + w + x = 4 + 2 + 1 = 7. When this is subtracted from 7, the value is 0.

The group category value is `2` because r + x = 4 + 1 = 5. When this is subtracted from 7, the value is 2.

The other category value is `7` because no permissions = 0. When this is subtracted from 7, the value is 7.

For example:

```
umask 000        Set default to allow full access to everyone
   touch test.000   Create file test.000 showing the resulting
 permissions
   umask 017
   touch test.017
   umask 022
   touch test.022
   umask 027
   touch test.027
   umask 077
   touch test.077   Allows access to only the owner
   ls -l test.*     (execute access (x) does not display for non-
executables)
     -rw-rw-rw-   Jun 12 12:04 test.000
     -rw-rw----   Jun 12 12:05 test.017
     -rw-r--r--   Jun 12 12:05 test.022
     -rw-r-----   Jun 12 12:05 test.027
     -rw-------   Jun 12 12:05 test.077
```

It is important to note that files created by the jobsub process are not owned by the same user as the user for whom the process is being run. For example, if SAISUSR submits GLOLETT then `glolett_12345.log` and `glotlett_12345.lis` will be owned by the account running jobsub, *not* SAISUSR. If reports are run to the database and viewed by GJIREVO, then this may not be an issue. However, if the reports needs to be accessed on the server then this may be a concern.

To determine which group a user is in, use this command: `id <username>`

For example:

```
id user001
```

```
uid=6356(user001) gid=401(banner
```

Windows platform

Several scripts are used in the submission of jobs on Windows.

Additionally, job submission makes use of a perl module, `sctban.pm`, sometimes referred to as an include file, to perform many common tasks such setting up the environment and printing output. All global subroutines contained in `sctban.pm` are prefixed by sctban, and all global variables set are prefixed by sctban. If you write any perl scripts of your own and want to use the subroutines contained in `sctban.pm`, you must include a use sctban statement before using any of the common routines or runtime errors will result. This should be followed immediately with:

```
&sctban_determine_os; &sctban_os_specific_env;
```

For an example, please refer to the section that describes gurjobs.pl.

Initially, a perl script called gjajobs.pl is started by the system function call in GURJOBS. gjajobs.pl establishes an execution environment by calling the `sctban_os_specific_env` function. How the values for the variables are obtained is controlled by the `BANENV` environment variable. `BANENV` is set from the **Control Panel** > **System** > **Environment**. A value of `REG` for `BANENV` indicates to set variables based on their value in the System Registry. A value of `ENV` means to use the value currently assigned to the environment variable first, and, if not set, default the value from the registry.

Next, the following variables are established:

```
$sctban_process_name $ARGV[0]

$sctban_process_type $ARGV[1]

$sctban_user_id $ARGV[2]

$sctban_password $ARGV[3]

$sctban_oneup_number $ARGV[4]

$sctban_printer_name $ARGV[5]

$sctban_form $ARGV[6]

$sctban_submit_time $ARGV[7]
```

Note: These are the arguments passed to gjajobs.pl as described above in the section on the GUQINTF page.

The `sctban_jsub_env` function is called to set additional environment variables followed by a call to `sctban_os_specific_jsub` to invoke `gjawnts.pl`. This script opens up a background Windows process and calls `gjajsub.pl` which actually constructs the command to run the job based on `sctban_process_type`. Output from the execution is saved into temporary files which are assembled into a single file after the job is run. These files are put in the directory specified by `banner_jobsub_home`. The naming convention is:

```
jobsub_home_sid_user_processname_oneupno
```

Batch Java scripts

The Java based Banner job submission processes need to use scripts to run. There is a certain setup that needs to be done within the scripts for the Java code to run. These scripts are shipped with setup that works for most clients.

However, changes to various application scripts were required for the process to run on a specific environment such as 64-bit. These changes were required for each and every one of the scripts, one per object/JAVA process. Whenever a new release was installed and these scripts are redelivered, the clients have to make these changes manually.

These modifications have been centralized to a General owned script instead of product or object level scripts. A new script has been created that enables the setting of Java environment variables for use for batch process. These scripts enable clients to define their specific environment in one script that will then be called by multiple batch processing scripts. This eliminates the need to update every Java based script after ever install.

The following setups have been centralized:

- Oracle connection string - This is used by the JAVA process to make connection to the Oracle database.
- UNIX ONLY - Path to where the Java Virtual Machine (JVM) is located. Sets the LD_LIBRARY_PATH which is need by UNIX to run the JVM.
- Class path to where the connection libraries are located. Each version of the JVM uses a different set of libraries to make connection to Oracle database. This communicates to the JVM where these libraries can be found.

The following new scripts have been delivered for each of the various environments:

- UNIX - banjavaenv.shl
- Windows - banjavaenv.pm

These scripts are located at `<BANNER_HOME>/general/misc` for UNIX/LINUX/NT.

Job Submission processing

The Job Submission Profile Maintenance (GJAJPRF) page is used to maintain Operating System specific information for your user ID. The base table for the page is the Personal Preference Table (GURUPRF).

All entries shown on this page have an internal identification of JOBSUB, contained in the `GURUPRF_GROUP` column. The entries on the page identify in which sub process this transaction will be used. The Value has several meanings based on the value of the JOBSUB Component.

There are currently three JOBSUB components that can be manipulated using this page: `DEFAULT_PRINTER`, `LOCAL_DIRECTORY`, and `GURJOBS_DIRECTORY`. These three details will be stored with other personal user preferences on GURUPRF.

The `DEFAULT_PRINTER` and `LOCAL_DIRECTORY` are created automatically when you print and save (respectively) from the Job Submission Review Output (GJIREVO) page.

The `LOCAL_DIRECTORY` can be any directory name that you can write to from your PC.

TECHNICAL NOTE: The GURJOBS program provides two mechanisms to specify the location of output files generated from batch jobs. One technique involves looking up the username; the other allows for the specification of a directory name to be passed to GURJOBS.

Look up the user

For UNIX, this is done be reading the `/etc/passwd` file. If the userid is found in the file, the literal `jobsub` is appended to this directory, and, if the directory exists and can be written to, this directory is substituted for the HOME environment variable before the `HOST` command executes.

About this task

If the directory does not exist, or it can not be written to, the current `HOME` directory is used when the job is run.

Procedure

1. Logon as the user `SYSTEM`.
2. Change to the SYS$SYSTEM directory by entering:

   ```
   SET DEF SYS$SYSTEM
   ```

   This should be the location of the sysuaf.dat file.
3. Run the authorize utility by entering:

   ```
   RUN AUTHORIZE
   ```

   The command prompt will change to UAF>.
4. Generate the sysuaf.lis file by entering the word `LIST`.UAF> LIST.
5. To exit the authorize utility enter the word `EXIT`.UAF> EXIT.

Results

The sysuaf.lis file will contain information about all logons for the machine. A sample follows.

Owner	*Username*	*UIC*	*Account*	*Privs*	*Pri*	*Directory*
`Banner7`	`BANNER7`	`[522,0]`	`BANNER`	`All`	`4`	`$DISK1:[BAN71_ROOT]`

When GURJOBS looks up a user, it opens this file, so it must be copied to a directory that can be accessed by the GURJOBS program while it's running. Usually the sys$login directory will suffice. Also make sure the privileges on the sysauf.lis file are changed if needed so it can be opened by the userid used to submit GURJOBS.

Alternatively, you could change GURJOBS so that the location of the sysaf.lis file was fully-qualified.

If the userid is found, a temporary .com file is built and a set def to this directory is written to the file, followed by the HOST command. If the directory is not found, the HOST command is issued directly.

Note: Your printer destination is controlled by your host login.

The `DEFAULT_PRINTER` must first be established on the GTVPRNT page as a valid printer code.

Pressing the Insert Record key will create the `GURJOBS_DIRECTORY` row. This preference is used to specify the name of a directory where output from Pro*C jobs will be placed when the job is run from the Process Submission Control (GJAPCTL) page.

Specifying a home directory

The GJAJPRF page may be used to specify the location of a *home* directory to be used when batch jobs are run. The `GURJOBS_DIRECTORY` preference indicates this, and is stored in the database table GURUPRF.

There are several ways to specify a value:

- If the record does not exist, select the create record key. The page will create a `GURJOBS_DIRECTORY` preference and attempt to find a home by sending a request to GURJOBS to lookup the username as described above. If found, a directory name is returned.
- You may also enter the name of a directory. (For example, a file system that gets exported and mounted to a PC.) This will then be sent to GURJOBS and an attempt will be made to create a test file in this location. If successful, the name will be accepted. If not, an error message will be issued.
- You can also enter the literal `LOOKUP`, which will send a request to GURJOBS to lookup the user as described above.
- You can enter the literal `DATABASE`. This option is valid for Pro*C programs and General COBOL programs, and will cause the output to be placed in the database. It can subsequently be reviewed on the GJIREVO page.

  Note: The Insert Output Program (GURINSO) is a Pro*C program that is used to insert the output into the database if the literal `DATABASE` had been entered in the Printer field of the Process Submission Control (GJAPCTL) page. The gjajobs.shl/.com file invokes execution of the program. If the program runs through the invocation of a command procedure, such as GLBDATA, the command procedure will invoke GURINSO.

The Saved Output Review (GJIREVO) page provides the ability to save the output as a file in a directory and the ability to print the output if a network printer is available to the user. When a request to print the output is made, the output is first saved to a local file and then printed by issuing a copy command to the printer specified on the GTVPRNT page. You can also purge the output from the database using this Global variables.

Note: No attempt is made to delete the file from the `LOCAL_DIRECTORY` if a save or print operation was performed.

Using Job Submission

Before attempting to print any data with the Saved Output Review page, check that a printer has been set up on the Printer Validation (GTVPRNT) page.

To produce the output, run a job in Job Submission as usual but be sure to put the word `DATABASE` in the **Printer** field of the Process Submission Control (GJAPCTL) page.

Note: If you run multiple jobs with the same name, the system should not overwrite the existing output because Job Submission incorporates the job's one-up number as part of the generated file name. If you are running jobs of some other type that do not use the one-up number as part of the file name, you may overwrite an existing file.

To view and print the output you created, access the Saved Output Review (GJIREVO) page.

Enter a job name in the Job Name field or press the Job Name Button for a list of the jobs that were run under your user ID that have not been purged from the database. You can double-click to select the desired output.

Your output will immediately appear in the Saved Output block of the page for review. At this point you can select the **Save and Print** button to save your output to your local directory, and print a copy of the output to the printer you specify.

Select the **Save to File** button to save your output to the your local directory. Select the **Delete Output** button to remove the selected file from the database.

Note: No attempt will be made to delete the file from your local directory after you have saved it. Local directory maintenance of files is up to the individual site using this procedure.

GURJOBS

GURJOBS is a PRO*C program created to handle the passing of jobs on a system network. It receives messages sent by the `Guqintf_Pipeit()` method in the GUQINTF page, on a `ORACLE PIPE` named GURJOBS.

When it receives a message, it must unpack it to determine what course of action to take. This is indicated by the first message, which is the request type.

However, if the Advanced Queuing alternate communication mechanism has been implemented (an alternative to `DBMS_PIPE`), GURJOBS instead listens and dequeues messages from queue `GURJOBS_Q`. These messages are sent (or enqueued) by the `GuqintServices.java.aqit()` method in the GUQINTF page. After dequeuing the message, GURJOBS inspects message fragment `MF_01` (see object type `g_msg_fragments`) which is the request type.

Currently, GURJOBS is designed to process three types of requests:

1. `HOST requests` - usually those originating from the GJAPCTL page. These jobs are submitted into the background (where available) and control is returned immediately to GUQINTF.
2. `WAIT requests` - typically initiated from an application page. GLRVRBL is an example of a WAIT type. GURJOBS waits for the request to be fulfilled (if it can) before sending a response back.
3. `EXIT requests` - terminate GURJOBS. The exit command is sent by signing on to SQL*Plus and starting the gurstop.sql file contained in the general/plus subdirectory.

Processing with DBMS_PIPE

If using the DBMS_PIPE communication mechanism, processing proceeds as follows.

The second message unpacked is the HOST command built by the GUQINTF page.

The third message names a return pipe. The name of the return pipe is generated by executing the `DBMS_PIPE.UNIQUE_SESSION_NAME` function which returns an unique name based on the connection to the database, much like `USERENV('SESSIONID')`.

The fourth message is optional and will only be present if a directory name has been established on the GJAJPRF page. If a directory name was not provided, GURJOBS will attempt to look it up by extracting the username from the command.

GURJOBS takes the unpacked message and issues the “system” function passing the host command as its argument.

```
system(command_string);
```

It then packs a message saying that the request is being processed and sends it back to the `Guqintf_Pipeit()` method. This message is relayed back to GJAPCTL, which displays it on the status line of the page. These messages are not usually displayed by application because they typically use the GJBRSLT table to indicate the status of the run.

When the system function is executed it will either execute the GJAJOBS file or the name of the file specified in the command name field on the GJAJOBS page.

Processing with DBMS_PIPE

If using the Advanced Queuing alternate communication mechanism (an alternative to `DBMS_PIPE`), processing proceeds as follows.

Message fragment `MF_02` holds the `HOST` command built by the GUQINTF page. This message fragment carries sensitive data.

Note: GURJOBS_Q queue messages may contain sensitive data needed to run jobs. These messages are persisted and therefore, the sensitive portion of these messages (MF_02) is encrypted. The GURJOBS process decrypts the sensitive portion. Database access to the decryption package (GSPCRPU) can be restricted but must, at minimum, be accessible to the GURJOBS process (the user running the GURJOBS process).

Message fragment `MF_03` is optional and will only carry a value if a directory name has been established on the GJAJPRF page. If a directory name was not provided, GURJOBS will attempt to look it up by extracting the username from the command.

Message fragment `MF_MISC_01` holds a unique token value. This value is established in the `GuqintServices.java.aqit()` method in the GUQINTF page. This page passes the unique token value, through the queue `GURJOBS_Q`, to the GURJOBS process. The page then listens (a conditional dequeue operation) for this unique token value on queue `GURJOBS_RTN_Q`.

GURJOBS takes the dequeued message and issues the “system” function passing the host command as its argument system (`command_string`); it then enqueues a message on return queue `GURJOBS_RTN_Q`. This queue message holds the unique token value that was previously obtained off of the `GURJOBS_Q` queue message. The return message indicates that the request is being processed and sends this back to the `AQIT` pl/sql unit. This message is relayed back to GJAPCTL, which displays it on the status line of the page. These messages are not usually displayed by application because they typically use the GJBRSLT table to indicate the status of the run.

When the system function is executed it will either execute the GJAJOBS file or the name of the file specified in the command name field on the GJAJOBS page.

IDLEWAIT timeout configuration modification for GURJOBS.pc

The maximum wait time waiting for a message on either the GURJOBS pipe or the `GURJOBS_Q` queue was 345,600 seconds or four days (86,400 seconds per day). The GURJOBS process stopped if it was idle for four days.

After the 8.3 release, the `gurjobs.pc` process has been modified such that the wait time (for sitting idle) is no longer hard coded at 345,600 seconds (4 days). The wait time is externally configurable now.

This modification reads IDLEWAIT timeout from gtvsdax and will only time out if it is idle for that number of seconds. The `max_wait_receive`, which was previously hard valued to 345,600 seconds (4 days), is now obtained from the gtvsdax row using the function `get_GtvsdaxWaitSeconds()`. This gtvsdax row is delivered with a value of 345,600 seconds and a value of 86,400,000 seconds (1000 days) is the maximum.

Note: Oracle Development has confirmed (November-2010) that 21,474,836 is the upper limit if a number is specified for wait time during a dequeue operation. Therefore, it is recommended that, if you are using the GURJOBS_Q queue aspects of GURJOBS.pc processing and are looking to use a larger IDLEWAIT timeout value than that which is delivered (345,600 seconds or 4 days), then you should use a value that is less than or equal to 21,474,836 seconds (approximately 248.5 days).

Managing Job Submission on Windows

This section provides information on running the job submission `GURJOBS.PC` process on Windows.

Starting Job Submission for your default database

To start the Job Submission Application Server (GURJOBS) program you will need to perform the following steps.

Prerequisites

This assumes that you have your `ORACLE_SID` entry in the Oracle Registry set to your initial Banner install database, usually `SEED`.

About this task

This also assumes that your System Environment variable `BANENV` is set to `REG`. `REG` means that the initially installed Banner key will be used from the registry. To view and change the `BANENV` variable, select Control Panel > System>Environment.

To start GURJOBS on Windows, do the following:

Procedure

1. Check the value of `ORACLE_SID` in the Oracle Registry.
2. Check the value of the System Environment variable `BANENV`.
3. Position in the \general\misc directory under Banner's Home directory.
4. Start the Perl script gurjwnt.pl. This will start job submission in the background. Note the space between the userid and the password.

   ```
   perl gurjwnt.pl <uid> <passwd>
   ```
5. Bring up the NT task manager and verify that gurjobs.exe is running.

   To keep job submission running on NT you must leave the administrator account logged in on the console. The console can be locked so that a password is needed to access it, but it is still logged in.

   In the future, instructions will be published for how to run job submission as a service. This, will require some files from the NT resource kit.

Starting Job Submission for multiple databases

The below example assumes you will have a SEED and TRNG instance - case does NOT matter.

Procedure

1. Change the `BANENV` setting to `ENV` (for ENVironment) in the System Environment. Select Control Panel>System>Environment to change this value. Changing `BANENV` to be `ENV` will allow Banner to override a registry entry with a value from the environment.
2. Create a LOCAL directory (optional - this could all be done in the Banner directories). Copy the `general\misc\gurjwnt.pl` script into the LOCAL directory to a name of `gurjwnt_seed.pl`.
3. Edit the gurjwnt_seed.pl script and add a line following the line `&sctban_os_specific_env;` to set the `ORACLE_SID` for this instance as follows:

   ```
   $ENV{"ORACLE_SID"} = "SEED";
   ```
4. Save and exit the script.
5. Run this script to start GURJOBS against the SEED database. Substitute your actual path for `c:\local\`.

   ```
   perl -S c:\local\gurjwnt_seed.pl <uid> <pswd>
   ```

   Do the same tasks for `GURJWNT_TRNG.PL`, substituting `TRNG` for `SEED`.

Managing Job Submission on UNIX

This section provides information on running the job submission `GURJOBS.PC` process on Unix.

Starting Job Submission for your default database

Create a new account for every `ORACLE_SID` to run GURJOBS or SFRPIPE processes.

Note: The gurjobs owner is not an interactive account. Ensure no prompts for database SID are displayed while logging into the Job Sub Unix account.

For example, Unix ID `banjobs` has a `$HOME` directory of `/u01/banner/banjobs`.

Note: The text included in this example can be used in new files created in the BANJOBS `$HOME` directory.

Listing All BANJOBS .Profile

```
#.profile
#An example listing of the banjobs' .profile
export ORACLE_BASE=/u02/oracle
export ORA_NLS10=$ORACLE_HOME/nls/data
export TNS_ADMIN=$ORACLE_BASE/local/network
export LD_LIBRARY_PATH=/u01/cobol/lib
SID BANNER_HOME JOBSUB_ACCOUNT JOBSUB_HOME (SYS$LOGIN)
BAN7 a20:[sct.ban7] jobsub7x a20:[sct.jobsub.ban7]
BAN8 a20:[sct.ban8] jobsub8x a20:[sct.jobsub.ban8]
PROD a20:[sct.prod] jobsub_prod a20:[sct.jobsub.prod]
TEST a20:[sct.test] jobsub_test a20:[sct.jobsub.test]
Banner General Technical Reference Manual | Reports and Processes 178
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
export JDK=$ORACLE_HOME/jdk/jre/lib/i386 #(operating system specific)
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$JDK/native_threads #(operating
system specific)
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$JDK/server:$JDK #(operating
system specific)
export TWO_TASK=unix1_prod
export BANNER_HOME=/u01/bannner
export DATA_HOME=$BANNER_HOME/dataload
export COBPREF='perl /u01/banner/links/banfjsv.pl ' #(specific to
 Fujitsu
NetCOBOL)
export EXE_HOME=$BANNER_HOME/general/exe
export BANNER_LINKS=$BANNER_HOME/links
export ORACLE_PATH=.:$BANNER_LINKS
export PATH=$PATH:$EXE_HOME:$BANNER_LINKS
```

Starting GURJOBS or SFRPIPE from a CRON

```
:
# banjobs_driver.shl
# This script may be run as banjobs from a cron to start gurjobs/
sfrpipe.
# You may need to give banjobs permissions to run cron jobs (/etc/
cron.d/
cron.allow).
# This script may also be run from the Unix command prompt.
LOGFILE1=/u01/banner/banjobs/gurjobs.log;export LOGFILE1
LOGFILE2=/u01/banner/banjobs/sfrpipe.log;export LOGFILE2
/u01/banner/banjobs/gurjobs.shl >>${LOGFILE1} 2>&1 &
/u01/banner/banjobs/sfrpipe.shl >>${LOGFILE2} 2>&1 &
```

Starting GURJOBS by calling BANJOBS_DRIVER.SHL

```
:
# gurjobs.shl
# This script is called by banjobs_driver.shl to start gurjobs.
(gurjobs -o jobs1 >>/u01/banner/banjobs/gurjobs.log 2>&1) << endofit
saisusr
u_pick_it
endofit
```

Starting SFRPIPES by calling BANJOBS_DRIVER.SHL

```
:
# sfrpipe.shl
# This script is called by banjobs_driver.shl to start sfrpipes.
cd /u01/banner/banjobs
(sfrpini >>/u01/banner/banjobs/sfrpipi.log 2>&1) << endofit
saisusr
u_pick_it
endofit
```

Starting BANJOBS_DRIVER.SHL from the UNIX prompt or from a CRON

The banjobs_driver.shl script can be started at the Unix prompt or from a cron. Before starting, create empty log file.

```
su - banjobs
touch gurjobs.log
touch sfrpipe.log
su
cd $BANNER_HOME/jobsub
./banjobs_driver.shl
```

Determining if GURJOBS or SFRPIPE is running in background

To determine if gurjobs/sfrpipe is running in the background, execute the following:

```
ps -efl | grep -i gurjobs
ps -efl | grep -i sfrpipe
```

Stoping GURJOBS using the Banner Baseline script GURSTOP.SQL

To stop gurjobs, use the Banner baseline script gurstop.sql, by executing the following:

```
sqlplus saisusr/u_pick_it @$BANNER_LINKS/gurstop.sql
```

Starting Job Submission for multiple databases

If there are five databases then have five separate `$BANNER_HOME` source trees and also have five separate UNIX JOBSUB accounts so that each account starts a copy of GURJOBS. This ensures everything is separate for GURJOBS and upgrades.

SID	BANNER_HOME	JOBSUB_ACCOUNT	JOBSUB_HOME (SYS $LOGIN)
PROD	/u01/prod	jobsub_prod	/u01/prod/jobsub
PPRD	/u02/pprd	jobsub_pprd	/u02/pprd/jobsub
TRNG	/u03/trng	jobsub_trng	/u03/trng/jobsub
TEST	/u04/test	jobsub_test	/u04/test/jobsub
SEED	/u05/seed	jobsub_seed	/u05/seed/jobsub

Now you have five UNIX JOBSUB accounts defined above and each would have a separate default profile (`.profile` or `.login`) and home directory.

After creating the profiles, start GURJOBS using these accounts. It defaults the environmental variables to that of ORACLE_SID (`banenv, oraenv`). This keeps all the environmental variables separate.

Managing Job Submission on non-database server

This section provides information on running the job submission `GURJOBS.PC` process on another Unix server besides the Unix database server for your Banner Administrative application.

Following are required for submitting a job on a non-database server:

- C compiler
- Cobol compiler
- Oracle Net (for example SQL*Net)
- Oracle Pro*C
- Oracle Pro*COBOL

  Note: For details on Oracle licensing contact your vendor. If you have a license for Oracle that was not issued by Oracle, contact your Account Manager.

- Java

For the Banner Pro*C programs (.pc) and Pro*COBOL programs (.pco) to be compiled on a non-database server, they should be copied to the non-database server.

The Banner `.shl` scripts should also reside in the links directory.

Typical directory structure

A typical directory structure for the Banner Home on Unix on the non-database server may look like the following.

```
cd $BANNER_HOME
ls -l
drwxr-xr-x 2 banjobs dba 11776 Jan 7 16:00 exe/
drwxr-xr-x 2 banjobs dba 512 Jan 2 15:24 jobsub/
drwxr-xr-x 2 banjobs dba 512 Jan 2 15:24 links/

drwxr-xr-x 2 banjobs dba 17408 Jan 3 12:47 general/c/
drwxr-xr-x 4 banjobs dba 13312 Jan 7 15:09 general/cob/
drwxr-xr-x 4 banjobs dba 13312 Jan 7 15:09 general/java/
```

Note: Additional product directories will exist for other installed products.

The Job Submission Unix ID in the example above is called `BANJOBS`. BANJOBS `$HOME` directory is `jobsub`. Files similar to the scripts described below reside in the jobsub directory.

Note: The non-database job sub server requires ICU to be installed to perform the c compilations.

Executing Banner Pro*C or Pro*Cobol programs

To execute Banner Pro*C or Pro*Cobol programs, you should execute the Banner GJAPCTL page and the executables from the non-database server. The output files will be created in the `/u01/ban_jobsub/jobsub` directory.

Job submission output

The output of a job may be viewed on the Saved Output Review (GJIREVO) page.

When you select **Options** > **Show Document (Save and Print File)**, the Job Submission Output either displays in a browser window or redirects for saving. If you select save, a file saves to the browsers designated "download" folder. From there you can print the file.

Process PL/SQL packages with JOBSUB

A C process, gsubsql.pc has been developed as a wrapper to enable the submission of PL/SQL package jobs through jobsub. This will use standard Banner security to ensure that the users is authorized to submit the job.

About this task

The only requirement will be that your job will require its own .shl script, but other than that you will be able to submit SQL procedures through job submission.

To use this script, perform the following steps:

Procedure

1. On GJAJOBS, create an entry with PROCESS equal to the name of your PL/SQL package, a type of 'Procedure', and appropriate description and title.
2. On GSASECR, the OBJECTS section, create an entry with the name of your PL/SQL package, appropriate version number, and role (typically BAN_DEFAULT_M).
3. On GSASECR, assign the object just created to a user directly or thru a class (after adding the object to the class). You can even add the object to a security group on GSADSEC if desired.
4. Create a script with the same name as the name of your PL/SQL package.
5. Create a PL/SQL package with a procedure to perform the tasks needed. The package.procedure will be called with one parameter, the one_up_seqno.

   The 'process' must match the name of a PL/SQL package that contains the PL/SQL procedure that will be executed.

   The package.procedure will be called with one parameter, the one_up_seqno. The script created will need a statement similar to the following:

   gsubsql -n $ONE_UP -j $PROG -p name_of_PL/SQL_procedure $UIPW

Example

An example of this script being used is general/misc/gorsrin.shl and is included below.

```
:
#!/bin/sh
# gorsrin.shl - script to run the batch population selection program
. . .
#
LOGFILE=$LOG; export LOGFILE
DATE=`date "+%Y%m%d_%HH%MM"`; export DATE
PROG=gorsrin; export PROG
# Send all standard output and standard error to the logfile.
exec > $LOGFILE 2>&1
echo "Starting Shell Script for" $PROG
# Execute Proxy Access Common Matching People Load SQL script.
# gsubsql -n <one up number> -j <job name and name of package where
 procedure is located>
```

```
#         -p <name of procedure to execute> <user id/password>
# gsubsql will validate access and set roles based upon access to $PROG
 (which is the same as the gjapctl and script job name [gorsrin])
#   Then will call the procedure passing the one-up number as the only
 parameter. It is up to the procedure to
#   get the job parms from gjbprun as needed.
gsubsql -n $ONE_UP -j $PROG -p p_gen_proxyaccess_com_match $UIPW
SQLERROR=$?
echo "Ending Shell Script. Status=" $SQLERROR
echo " "
if [ $SQLERROR -ne 0 ] ; then
    echo "### +++ SQLPLUS failed: " $SQLERROR
    echo "Job terminated"
fi
# Load all files to database, if requested
#
if [ "$PRNT" = "DATABASE" ] ; then
  case $PSWD in
      /) UIPW=$PSWD ;;
      *) UIPW=$BANUID/$PSWD ;;
  esac
  echo "Loading files into the database for viewing on GJIREVO"
#
# The following find command will find all files with a name containing
 the process and one_up_seq except those with a file type
# of ".in" or ".shl" that were modified in the last 24 hours. It will
 list the name of the file, and then load it into the database
# for viewing on GJIREVO.
#
  find ${H}/${PROC}_$ONE_UP* ! \( -name "*.in" -o -name "*.shl" \) -
mtime -1 -exec ls -alt {} \; -exec gurinso -n $ONE_UP -l {} -j $PROG -w
 $BANUID $UIPW \;
fi
if [ $SQLERROR -ne 0 ] ; then
    exit $SQLERROR
fi
exit
```

Create a job to run a PL/SQL package.procedure thru jobssub

To create a job to run a PL/SQL package.procedure through job submission, perform the following steps.

Procedure

1. Define the PL/SQL test procedure as follows:

```
sqlplus system/manager
grant dba to general;
alter user general default role dba;
sqlplus general/u_pick_it
drop package body testsql;
drop package testsql;
DROP PUBLIC SYNONYM testsql;
CREATE OR REPLACE PACKAGE testsql AS
```

```
PROCEDURE testjob (one_up_no IN VARCHAR2);

END testsql;
/
SHOW ERRORS;
DROP PUBLIC SYNONYM testsql;
CREATE PUBLIC SYNONYM testsql FOR testsql;
CREATE OR REPLACE PACKAGE BODY testsql AS
PROCEDURE testjob (one_up_no IN VARCHAR2)
  IS
    cmd1         VARCHAR2(250);
    cmd2         VARCHAR2(250);
    result       VARCHAR2(250);
  BEGIN
    cmd1 := 'SELECT GJBPRUN_JOB FROM GJBPRUN WHERE
 GJBPRUN_NUMBER=''99'' AND GJBPRUN_ONE_UP_NO = '||one_up_no;
    cmd2 := 'UPDATE GJBPRUN SET GJBPRUN_LABEL = ''GSUBSQL COMPLETE''
 WHERE GJBPRUN_NUMBER=''99'' AND GJBPRUN_ONE_UP_NO = '||one_up_no;
--
    EXECUTE IMMEDIATE cmd1 INTO result;
    DBMS_OUTPUT.PUT_LINE('RESULT = '||result);
    EXECUTE IMMEDIATE cmd2;
--
  END testjob;
END testsql;
/
SHOW ERRORS;
sqlplus system/manager
revoke dba from general;
```

2. Test executing the procedure from SQLPLUS to confirm it is working as follows:

```
set serveroutput on
XECUTE testsql.testjob(1716);
```

The expected result is:

```
RESULT = TESTSQL
PL/SQL procedure successfully completed.
```

To continue the test, execute the following:

```
select GJBPRUN_LABEL from GJBPRUN WHERE GJBPRUN_NUMBER='99' AND
 GJBPRUN_ONE_UP_NO = 1716;
```

The expected result is:

```
GSUBSQL COMPLETE
```

3. Create the OS file `testsql.shl` in UNIX `or testsql.pl` in Windows in `$BANNER_HOME/general/misc`.

   a) In UNIX, the testsql.shl should contain the following:

```
:
#!/bin/sh
# testsql.shl
LOGFILE=$LOG; export LOGFILE
DATE=`date "+%Y%m%d_%HH%MM"`; export DATE
PROG=testsql; export PROG
exec > $LOGFILE 2>&1
echo "Starting Shell Script for" $PROG
gsubsql -n $ONE_UP -j $PROG -p testjob $UIPW
SQLERROR=$?
echo "Ending Shell Script. Status=" $SQLERROR
exit
```

   b) In Windows, the testsql.pl should contain the following:

```
# testsql.pl
use sctban;
&sctban_determine_os;
&sctban_os_specific_env;
&sctban_jsub_env;
$a1 = $ARGV[0];
$a2 = $ARGV[1];
$a3 = $ARGV[2];
$a4 = $ARGV[3];
$a5 = $ARGV[4];
$banner_exe  = $ENV{"BANNER_EXE"};
open (CPROCESS, "|${banner_exe}${sctban_dirsep}gsubsql
 -n ${a3} -j ${a4} -p testjob ${sctban_user_pass} >
${sctban_file_name}.stdout 2>${sctban_file_name}.stderr");
close (CPROCESS);
&sctban_rebuild_log;
```

4. Login to Banner and setup TESTSQL process, by defining the job on GJAJOBS as follows:

```
Process = TESTSQL
Title   = Testing PL/SQL using GSUBSQL
System  = G
Type    = Procedure
```

   On the Objects section of GSASECR, define TESTSQL as follows:

```
TESTSQL  8.5  G  BAN_DEFAULT_M  PUBLIC
```

   On the Users section of GSASECR, assign TESTSQL to userid for testing.

5. Login as userid and run TESTSQL from GJAPCTL. It should create a log file named `rocoram2_ban8_saisusr_testsql_1717.log` that contains the following:

   ```
   Starting gsubsql (Release 8.5)
   Connected.
   Running
   select GJBPRUN_LABEL from GJBPRUN WHERE GJBPRUN_NUMBER='99' AND
    GJBPRUN_ONE_UP_NO = 1719;
   ```

   When complete and the PL/SQL procedure ran correctly through JOBSUB and updated the database, the following message will display:

   ```
   GSUBSQL COMPLETE
   ```

APIs

Application Programming Interfaces (APIs) facilitate the integration of Banner with other applications on campus and simplify code by encapsulating business logic in database packages.

An API is a central program that creates, updates, and deletes data. APIs also execute and validate business rules before inserting or updating information.

Detailed documentation for APIs can be downloaded from the Customer Support Center. Select “API Documentation” when browsing for product documentation. There is also an optional API/ERD Index (`api_erd_index_guide.zip`) that provides a single starting point for HTML-based API documentation and Entity Relationship Diagrams.

APIs used in Banner General

The following tables and pages use APIs to process data in Banner General. The page listed next to the table in this chart is the representative source used to build the API validation and business rules. The APIs replace the corresponding code in the Banner pages.

Most of the APIs support create, update, and delete signatures. Exceptions, such as queries, are noted under Task Performed.

Table	Page	API Object Name	API Entity Name	Task Performed
GORFGBP	GOAFGAC	gb_bus_prof_rule	BUSINESS PROFILE RULE	Assigns the categories of users and their privileges to each domain and predicate for a FGAC VBS Group Rule.
GORFBPR	GOAFBPR	gb_businessprofile	BUSINESS PROFILE	Groups users with similar responsibilities for FGAC VBS processing.
GORFBPI	GOAFBPI	gb_busprofpii	BUSINESS PROFILE PII	Groups users with similar responsibilities for FGAC PII processing.
GORCMDD	GORCMDD	gb_cm_data_dictionary	COMMON MATCHING DATA DICTIONARY	Maintains data dictionary entries for common matching.

Table	Page	API Object Name	API Entity Name	Task Performed
GORCMDH	GORCMDH	gb_cm_disp_hier	COMMON MATCHING DISPLAY HIERARCHY	Maintains display hierarchy information for common matching.
GORCMDO	GORCMRL	gb_cm_display_options	COMMON MATCHING DISPLAY OPTIONS	Maintains options for how common matching search results will be displayed.
GORCMSR	GORCMRL	gb_cm_rules	COMMON MATCHING RULES	Maintains common matching rules.
GORCMSP	GORCMRL	gb_cm_source_priority	COMMON MATCHING SOURCE PRIORITY	Maintains priority numbers for common matching source codes.
GORCMSC	GORCMSC	gb_cm_source_rules	COMMON MATCHING SOURCE RULES	Maintains source codes for common matching.
GORCMUP	GORCMRL	gb_cm_user_procedure	COMMON MATCHING USER PROCEDURE	Associates packaged procedures to be used by the common matching procedure.
GOBCMUS	GORCMUS	gb_cm_user_setup	COMMON MATCHING USER SETUP	Maintains users for common matching.
GTVCURR	GTVCURR	gb_currency	CURRENCY	Maintains currency codes.
GURCURR	GUACURR	gb_currency_rate	CURRENCY RATE	Maintains rates for currency conversion.
GORDMCL	GORDMCL	gb_displaycolumn	DISPLAY COLUMN	Maintains data items (columns) for Protection of Sensitive Information security.

Table	Page	API Object Name	API Entity Name	Task Performed
GORDMSK	GORDMSK	gb_displaymask	DISPLAY MASK COLUMN RULE	Maintains rules for users and items (columns) for Protection of Sensitive Information security.
GOBFDMN	GORFDMN	gb_domains	DOMAIN	Maintains domains for FGAC processing.
GOBFDTP	GORFDTP	gb_domaintype	DOMAIN TYPE	Maintains domain types for FGAC processing.
GOREMAL	GOAEMAL	gb_email	EMAIL	Maintains e-mail address records.
GORFGUS	GOAFGAC	gb_fgac_user_rule	FGAC USER RULE	Assigns users and their privileges to each domain-predicate combination for a FGAC VBS Group Rule.
GOBFEOB	GORFEOB	gb_fgacexcluded-objects	FGAC EXCLUDED OBJECT	Maintains objects excluded from all FGAC processing.
GOBFGAC	GOAFGAC	gb_group_rules	GROUP RULE	Maintains the Active and Effective Date characteristics for FGAC VBS group rules.
GORIMMU	GORIMMU	gb_immunization	IMMUNIZATION	Maintains immunization status information.
GORIROL	—	gb_institution_role	INSTITUTION ROLE	Calculates institutional roles.
GORICCR	GORICCR	gb_integ_config	INTEGRATION CONFIGURATION	Maintains integration configuration settings.

Table	Page	API Object Name	API Entity Name	Task Performed
GORBLOB	TGISTMT	gb_large_object	LARGE OBJECT	Provides centralized storage for large objects, including graphics files and PDFs.
GORNAME	GORNAME	gb_name_translate	NAME TRANSLATE	Maintains aliases or nicknames associated to given names (No update).
GORNPNM	—	gb_np_name_trans	NP NAME TRANS	Maintains aliases or nicknames associated with non-person names.
GORPRAC	GOQCLIB	gb_person_race	PERSON RACE	Maintains person race data.
GORFDPI	GORFDPI	gb_pii_tables	PII TABLE	Maintains table rules for FGAC PII processing.
GORINTG	GORINTG	gb_partner_rule	PARTNER RULE	Maintains integration partner system rules.
GOBANSR	GOATPAD	gb_pin_answer	PIN ANSWER	Stores answers to security questions and compares the answers provided by users during the PIN reset process.
GOBQSTN	GOAQSTN	gb_pin_question	PIN QUESTION	Stores and retrieves security questions used in the PIN reset process.
GORADID	%IDEN pages	gb_additional_ ident	ADDITIONAL IDENT	Stores and retrieves additional ID information.
GORRACE	GORRACE	gb_race_ethnicity	RACE ETHNICITY	Maintains race rule information.

Table	Page	API Object Name	API Entity Name	Task Performed
GORSDDV	GOADISC	gb_sde_discrim_value	DISCRIM VALUE	Stores and retrieves discriminator values for SDE processing.
GOBSDDC	GOADISC	gb_sde_discriminator	SDE DISCRIMINATOR	Stores and retrieves discriminator information for SDE processing.
GORSDAM	GOASDMD	gb_sde_metadata	SDE METADATA	Stores and retrieves metadata for SDE attributes.
GOBSDTB		gb_sde_table	SDE TABLE	Stores and retrieves information on Banner tables that have been extended with supplemental data through SDE.
GOBTPAC	GOATPAC	gb_third_party_access	THIRD PARTY ACCESS	Maintains cross-references between third party system user IDs and Oracle user ID (No delete).
GOBFPUD	GOAFPUD	gb_userdefault	USER DEFAULT	Defines a user's home domain for FGAC PII processing.
GORFPII	GOAFPII	gb_userpiidomains	USER PII DOMAIN	Maintains domains for each user for FGAC PII processing.
GORFPRD	GORFDTP	gb_vbs_predicate	VBS PREDICATE	Maintains predicate statement (WHERE clause) for FGAC VBS processing.

Table	Page	API Object Name	API Entity Name	Task Performed
GORFDPL	GORFDPL	gb_vbs_tables	VBS TABLE	Maintains tables associated with each domain for FGAC VBS processing.
GORVISA	GOAINTL	gb_visa	VISA	Maintains visa codes.
GURFWWK	GUAFWWK	gb_flex_wrk_week	FLEX_WRK_WEEK	Maintains flexible work week data.
GURUPPP	9x Admin Pages	gb_useruserprefs	USER_USERPREFS	Maintains User Preferences (UPA) for Admin.
GURMPPP	9x Admin Pages	gb_mstruserprefs	MSTR_USERPREFS	Maintains User Preferences (UPA) for Admin master preference data.

APIs used in Banner General with Student pages and tables

The following Student tables and pages use APIs to process data in Banner General and Banner Student.

Table	Page	API Object Name	API Entity Name	Task Performed
SPRADDR	SPAIDEN	gb_address	ADDRESS	Maintains address information.
SPBPERS	SPAPERS	gb_bio	BIO	Maintains biographic/ demographic information for an individual.
SLBBLDG	SLABLDG	gb_bldgdefinition	BLDGDEFINITION	Maintains building information.
SSRMEET	SSASECT	gb_classtimes	CLASSTIMES	Maintains section and event meeting times.
SPREMRG	SPAEMRG	gb_emergency_ contact	EMERGENCY CONTACT	Maintains emergency contact information for an individual.

Table	Page	API Object Name	API Entity Name	Task Performed
SPRHOLD	SOAHOLD	gb_hold	HOLD	Places or removes holds on an account.
SPRIDEN	SPAIDEN	gb_identification	IDENTIFICATION	Maintains person and non-person biographic/ demographic information.
—	—	gp_international_student	INTERNATIONAL STUDENT	Retrieves international student information from fsaATLAS.
SPRMEDI	SPRMEDI	gb_medical	MEDICAL CODE	Maintains information about medical conditions.
SORPCOL	SOAPCOL	gb_prior_college	PRIOR COLLEGE	Maintains a person's educational background information.
SORCONC	SOAPCOL	gb_pcol_concentration	PRIOR COLLEGE CONCENTRATION	Maintains educational background information on areas of concentration.
SORDEGR	SOAPCOL	gb_pcol_degree	PRIOR COLLEGE DEGREE	Maintains educational background information on degrees.
SORMAJR	SOAPCOL	gb_pcol_major	PRIOR COLLEGE MAJOR	Maintains educational background information on majors.
SORMINR	SOAPCOL	gb_pcol_minor	PRIOR COLLEGE MINOR	Maintains educational background information on minors.

Table	Page	API Object Name	API Entity Name	Task Performed
SLRRASG	SLARASG	gb_roomassignment	ROOMASSIGNMENT	Maintains dorm room assignments.
SLBRDEF	SLARDEF	gb_roomdefinition	ROOMDEFINITION	Maintains room information by building.
STVTERM	STVTERM	gb_stvterm	STVTERM	Queries term validation information.
SPRTELE	SPATELE	gb_telephone	TELEPHONE	Maintains telephone information.
		gp_cardholder		Process APIs related to campus card cardholders.
		gp_common_ matching		Checks for the existence of a record within the Banner System based on criteria (rules) defined for the source of the data.
		gp_entity_address		Process API to add or update address information about a person in the Banner system.
		gp_international_ student		Process API to retrieve International Student Information.
		gp_person_ identity		Process API to allow external systems to determine the unique identifier (ID number) for a person.

APIs for internal Banner operations

Please be advised that several Banner General APIs are currently intended only to support internal operations.

To ensure data integrity, these APIs are not supported when called by external applications or interfaces to manipulate data. The recommendation for external applications is to use message level integration to integrate with these entities in Banner.

The following APIs come under this disclaimer:

- `gb_advq_util`
- `gb_common`
- `gb_event`
- `gb_gtvcall`
- `gb_gtvccrd`
- `gb_gtvcelg`
- `gb_gtvcmsc`
- `gb_gtvcurr`
- `gb_gtvdadd`
- `gb_gtvdicd`
- `gb_gtvdiro`
- `gb_gtvdocm`
- `gb_gtvdprp`
- `gb_gtvdstp`
- `gb_gtvdunt`
- `gb_gtvemal`
- `gb_gtvemph`
- `gb_gtveqnm`
- `gb_gtveqpg`
- `gb_gtveqpm`
- `gb_gtveqts`
- `gb_gtvexpn`
- `gb_gtvfbpr`
- `gb_gtvfdmn`
- `gb_gtvfdtp`
- `gb_gtvfees`
- `gb_gtvinsm`
- `gb_gtvletr`

- gb_gtvlfst
- gb_gtvmail
- gb_gtvmenu
- gb_gtvmtyp
- gb_gtvntyp
- gb_gtvobjt
- gb_gtvpara
- gb_gtvpars
- gb_gtvpdir
- gb_gtvprnt
- gb_gtvproc
- gb_gtvptyp
- gb_gtvpurp
- gb_gtvrate
- gb_gtvrevn
- gb_gtvrrac
- gb_gtvrsvp
- gb_gtvrtng
- gb_gtvschs
- gb_gtvscod
- gb_gtvsdax
- gb_gtvsdiv
- gb_gtvsegc
- gb_gtvsqpa
- gb_gtvsqpr
- gb_gtvsqru
- gb_gtvsrce
- gb_gtvssfx
- gb_gtvsubj
- gb_gtvsvap
- gb_gtvsvba
- gb_gtvsvca
- gb_gtvsvcc
- gb_gtvsvcp
- gb_gtvsvcr
- gb_gtvsvdt
- gb_gtvsvel

- `gb_gtvsvep`
- `gb_gtvsvft`
- `gb_gtvsvgo`
- `gb_gtvsvio`
- `gb_gtvsvit`
- `gb_gtvsvpc`
- `gb_gtvsvrp`
- `gb_gtvsvtr`
- `gb_gtvsvts`
- `gb_gtvsysi`
- `gb_gtvtarg`
- `gb_gtvtask`
- `gb_gtvtrtp`
- `gb_gtvtsta`
- `gb_gtvttyp`
- `gb_gtvuoms`
- `gb_gtvutyp`
- `gb_gtvviss`
- `gb_gtvvpdi`
- `gb_gtvwfed`
- `gb_gubobjs`
- `gb_stvaccg`
- `gb_stvacyr`
- `gb_stvascd`
- `gb_stvasrc`
- `gb_stvasty`
- `gb_stvatyp`
- `gb_stvbchr`
- `gb_stvbldg`
- `gb_stvcamp`
- `gb_stvcipc`
- `gb_stvcitz`
- `gb_stvcnty`
- `gb_stvcoll`
- `gb_stvcomf`
- `gb_stvcoms`
- `gb_stvcomt`

- gb_stvdays
- gb_stvdegc
- gb_stvdept
- gb_stvdisa
- gb_stvempt
- gb_stvethn
- gb_stvetyp
- gb_stvfcnt
- gb_stvgeod
- gb_stvgeor
- gb_stvhldd
- gb_stvlang
- gb_stvlead
- gb_stvlevl
- gb_stvlgcy
- gb_stvmajr
- gb_stvmatl
- gb_stvmdeq
- gb_stvmrtl
- gb_stvnatn
- gb_stvorig
- gb_stvprcd
- gb_stvptyp
- gb_stvrdef
- gb_stvrelg
- gb_stvrelt
- gb_stvrmst
- gb_stvrrcd
- gb_stvsbgi
- gb_stvsite
- gb_stvspon
- gb_stvspsr
- gb_stvstat
- gb_stvsubj
- gb_stvtele
- gb_stvtrmt
- gb_xml_generator

Interfaces

This section discusses the interfaces with external user systems and the interfaces within Banner.

Interfaces with external user systems

The following are the interfaces with external user systems.

GURFEED table

This table contains financial transactions from Banner applications which are to be processed by the client's Accounting system through a user interface program.

GURAPAY table

This table contains single line invoices from Banner applications which are to be processed by the client's Accounts Payable system through a user interface program.

Interfaces within Banner

The following are the interfaces within Banner.

GURFEED table

This table contains financial transactions from other Banner applications or client-developed applications which are to be processed into Banner Finance using the FURFEED and FGRTRNI processes.

GURAPAY table

This table contains single line invoices from other Banner applications or client-developed applications, which are to be processed into Banner using the FURAPAY process.

COBOL compiling

This section provides general information about the COBOL compilation process, such as required directories, file locations, and example scripts.

Banner compile scripts are provided for new installations and upgrades to compile all code in the correct order. On UNIX machines the output from the compile will be placed by default in the `exe` subdirectory of the General product. If the compile routines for your port or site write into the current directory, then the output from the compiles will have to be migrated before they can be accessed by the users.

Note: If your compile procedure writes directly into the General product's `exe` subdirectory, this procedure must be run from an operating system account that has write permission into the target Banner directories.

Compiling COBOL under UNIX

Compiling COBOL code under the UNIX operating system is accomplished through the use of the `make` command, a special-purpose scripting language usually provided as part of the UNIX language development environment.

A makefile is needed for all but the most basic make operations; it specifies the actions to be taken to perform particular tasks, such as making an executable from a COBOL source file, or building an object file from a Pro*COBOL file.

To compile Banner Pro*COBOL code into executables, you must create a makefile that includes all of the proper options and libraries for the combination of operating system, Oracle, and compiler versions installed on your machine.

Create a Pro*COBOL makefile

The buildcob process is provided as a tool to assist Banner clients using UNIX systems in constructing a valid Pro*COBOL makefile for their particular operating system, Oracle, and compiler environment. Two files are provided: buildcob.c (source code for buildcob) and bancob.tpl (a Banner makefile template.)

About this task

To use buildcob, follow these steps:

Procedure

1. Log in as your Banner owner.
2. Make sure that your environment reflects the proper `ORACLE_HOME`. This is necessary because the `buildcob` process uses an Oracle demo makefile as a model and must be able to find it.

3. Enter the following:

   ```
   cd $BANNER_HOME/install
   ```

4. Compile the buildcob.c file with an ANSI-compliant C compiler. Some compilers require command-line parameters to recognize ANSI code; refer to your compiler documentation for details.

   Example: `cc buildcob.c -o buildcob`

5. Execute buildcob and respond to the on-screen messages and prompts.

   Note: The value for your COBOL compiler may differ from the default.

6. You should now have a makefile with the default name of sctprocb.mk in the `$BANNER_HOME/general/cob` directory. Use this makefile to compile a Banner Pro*COBOL program, and make changes to sctprocb.mk to resolve any errors. Sequent, NCR, and DEC Ultrix machines require that `COMP5=YES` be passed to the pre-compiler for byte storage compatibility. Other platforms may require that this be commented out.

   If you find that your local environment requires changes from the defaults, you may directly edit the provided bancob.tpl file so that your changes are preserved when you rerun the `buildcob` process in the future.

Example buildcob session

The following is an example dialog from a run of the `buildcob` process under the Digital UNIX operating system.

```
% cc buildcob.c -o buildcob
% buildcob
```

`buildcob` is a program that assists in the creation of a Banner Pro*COBOL makefile. Using a provided template and an Oracle makefile from your current release of the Oracle software, `buildcob` generates a new makefile that should work with your operating system and Pro*COBOL release.

Note: The generated makefile also includes comments to guide you in making manual changes if necessary.

You will now be prompted for information needed by the `buildcob` program. The default value for each option appears in parentheses after the prompt; if you want to accept the default for a particular option just hit enter. Each of these defaults is defined in a macro in the `buildcob` source code, making the setting of local defaults simple; comments in the source code explain the process.

Enter the name of your COBOL compiler; if your compiler is not present in your path, then you will need to specify a full directory reference.

```
e.g., /usr/local/cob
COBOL Compiler? ( cob )
```

Enter the name of the template file to be used to generate your new makefile. If you make local modifications to the provided template then you may want to copy the template to a different name and enter that name below.

```
SCT makefile template? ( bancob.tpl )
```

Enter the name of the model Oracle makefile. Oracle provides an example Pro*COBOL makefile that is used to make example programs and the Pro*COBOL executable itself; this file is scanned by `buildcob` to extract the proper library definitions for your system.

```
Oracle makefile model? ( /u02/app/oracle/product/9.2.0
       /precomp/demo/procob/procob.mk )
```

Enter the name of the new makefile that `buildcob` will generate; if a file by that name already exists it will be overwritten.

```
New makefile to create? ( /yy/banner/general/cob /sctprocb.mk )

bancob.tpl/yy/banner/general/cob/sctprocb.mkbuildcob terminated normally
Using sctprocb.mk
```

To use the `sctprocb.mk` makefile, position yourself to the directory containing the source code to be compiled, and enter a make command specifying both the makefile and the file to be generated.

```
make -f $BANNER_HOME/general/cob/sctprocb.mk PHPFEXP
```

Refer to your operating system documentation for further details on makefile construction and usage.

Reducing executable sizes

Pro*COBOL executables may be very large on some UNIX platforms, impacting both runtime performance and storage requirements.

To reduce the size of executables, you can use the Oracle Run Time System (rtsora) and compile your Pro*COBOL code to .gnt files, or you can use shared objects to provide dynamic linking at runtime.

Note: One, both, or neither of these methods may be available on your particular platform; refer to your Oracle and operating system documentation for further information.

To use the Oracle Run Time System, you must build an `rtsora` executable in your `$ORACLE_HOME/bin` directory using an Oracle-provided procedure. Banner supports the Oracle Run Time System by using two environment variables, COBPREF and COBSUFX, in all shell scripts that execute COBOL programs. These variables are created in cbanenv and banenv with null values; if you are using .gnt files, then COBPREF should be set to `rtsora` and COBSUFX to `.gnt`. The other alternative is to use dynamic linking, or shared objects. If your operating system and Oracle release support this option, then modify your copy of sctprocb.mk and add `–lclntsh` at the beginning of the LLIBS macro definition.

Example:

```
LLIBS=-lclntsh $(COBSQLINTF) $(LLIBSQL) $(TTLIBS)
```

Also, the environment variable `LD_LIBRARY_PATH` will need to be defined in order for the shared object references to be resolved at runtime.

Example:

```
LD_LIBRARY_PATH=/usr/lib/cob/coblib:/u02/app/oracle
        product/9.2.0/lib export LD_LIBRARY_PATH
```

Stripping your executables of debugging information may also significantly decrease their size; this can usually be done at compile time with the -s switch to the compiler, or later with the stand-alone strip program. Also, if you are using certain versions of SVR4-based operating systems (such as Dynix/ptx), the `mcs-d` command can be used to strip internal comments from the executables.

COBOL Compiling during Banner installation

For an initial installation of Banner, all products that have COBOL programs need to have them compiled. The Banner installation process uses the script banccob.shl on UNIX or banccob.com on OpenVMS to compile all COBOL code.

You may use the `.shl` (UNIX), or `.pl` (Windows) versions of the scripts below to compile a single product's COBOL code. The scripts are located in the appropriate product's /misc production directory.

Banner product COBOL compile procedures

The following is a list of the Banner products and their COBOL compile procedures.

Banner Product	Compile Procedures
A/R Module	tascmpl
Advancement	none
Finance	none
Financial Aid	rescmplx
General	gencmpl
INAS (where x equals the last digit of the aid year)	rescomplx
Payroll Module	paycmpl
Student	stucmpl

Executables are built in the `$BANNER_HOME/general/exe` directory. This directory must be in the PATH of each Banner user.

UNIX

Below is a sample of the Banner General COBOL compile shell script.

```
# gencmpl.shl
#
cd $BANNER_LINKS
#
make -f $BANNER_LINKS/sctprocb.mk GUAGETP.o
make -f $BANNER_LINKS/sctprocb.mk GUASETR.o \
CHECKOPT=”sqlcheck=full userid=baninst1/u_pick_it”
make -f $BANNER_LINKS/sctprocb.mk GUAVRFY \
BANOBJ=$BANNER_HOME/general/exe/GUAGETP.o
make -f $BANNER_LINKS/sctprocb.mk GLOLETT \
BANOBJ=$BANNER_HOME/general/exe/GUAGETP.o
make -f $BANNER_LINKS/sctprocb.mk GLBPARM \
BANOBJ=$BANNER_HOME/general/exe/GUAGETP.o
make -f $BANNER_LINKS/sctprocb.mk GLBDATA \
BANOBJ=$BANNER_HOME/general/exe/GUAGETP.o
make -f $BANNER_LINKS/sctprocb.mk GLBLSEL \
BANOBJ=$BANNER_HOME/general/exe/GUAGETP.o
```

Windows

Below is a sample of the Banner General COBOL compile command script.

```
use sctcomp;
$sctcomp_product_dir = “general”;
$sctcomp_input_file_ref = \*DATA;
&sctcomp_cobol_process;
_END_
GUAGETP.pco -exetype=obj
GUASETR.pco -exetype=obj -checkopt=full
GUAVRFY.pco
GLOLETT.pco
GLBPARM.pco
GLBDATA.pco
GLBLSEL.pco
```

C compiling

This section provides general information about the C compilation process, such as necessary directories, file locations, and example scripts.

Banner compile scripts are provided for new installations and upgrades to compile all code in the correct order. On UNIX machines the output from the compile will be placed by default in the `exe` subdirectory of the General product. If the compile routines for your port or site write into the current directory, the output from the compiles will have to be migrated before they can be accessed by the users.

Note: If your compile procedure writes directly into the General product's `exe` subdirectory, this procedure must be run from an operating system account that has write permission into the target Banner directories.

International Components for Unicode (ICU)

To enable the processing of UTF-8 characters in C processes, Banner has a dependency on the ICU library.

When you upgrade to a newer operating system version or a newer C compiler that does not support your current version of ICU, you must also upgrade ICU.

Related information

ICU - International Components for Unicode

Upgrade ICU for Unix based environments

Upgrade to ICU version 58.3 on a Unix-based Job Submission server environment.

Procedure

1. Log in to the JobSub server as root and stop the gurjobs process.
2. Enter `cd /usr/local/src` to change the directory.
3. Verify that file `icu4c-58_3-src.tgz` exists in the directory.
   You can download the file from the Ellucian Download Center.
4. Enter `mv icu icu_old` to move and rename the current `icu` directory.
5. Enter `gunzip -d < icu4c-58_3-src.tgz | tar xvf` – to unzip the file.

6. Change the directory to `cd /usr/local/src/icu/source` and enter the following command to make the file executable.

   ```
   chmod +x runConfigureICU configure install-sh
   ```

   Enter the appropriate command for your operating system to install ICU.

Operating System	Command
Linux	`./runConfigureICU Linux/gcc > icu_configure.log 2>&1`
HP-UX	`./runConfigureICU HP-UX/ACC --enable-64bit-libs > icu_configure.log 2>&1`
AIX	`./runConfigureICU AIX --with-library-bits=64 > icu_configure.log`
SOLARIS	`./runConfigureICU Solaris --with-library-bits=64 > icu_configure.log`

7. Verify that the *$ICU_HOME* and *$BANNER_HOME* environment variables exist. Establish these variables if they do not already exist.

   ```
   echo $ICU_HOME
   ```

   ```
   echo $BANNER_HOME
   ```

   The commands above return `/usr/local/src` and `/app/sghe/banner/SMPL` respectively.

8. Enter the following commands to clean, recompile and link files:

   - `gmake clean > icu_clean.log 2>&1`
   - `gmake > icu_make.log 2>&1`
   - `gmake install > icu_install.log 2>&1`

   Review the logs for errors.

9. Enter `ucon -V` to verify the ICU4C version installed.
   The command returns `uconv v2.1 ICU 58.3`
10. Log out of the server and log in again with as banner.
11. Enter `echo $BANNER_HOME` to verify the Banner home directory.
    The command returns `/app/sghe/banne/SMPL`.
12. Enter `cd $BANNER_HOME` to change the directory.

13. Copy the Banner specific file over the baseline Banner files compatible with the 58.3 ICU installation.

    Note: If running on an HP-UX operating system, you must copy the `tmcilib.cpp_hpux_583` file rather than the `tmcilib.cpp_583` file. You can do this by adjusting the `copyICU583_ban.shl` script to use this file, or you can manually copy the file after running the script.

    a) Enter `sh /general/misc/copyICU583_ban.shl >copyICU583_ban.log` to copy `tmcilib.cpp_583` over the previous ICU version.
    b) **Optional:** If you are running HP_UX, copy the `tmcilib.cpp_hpux_583` over the `tmcilib.cpp_583` file if you did not adjust the `copyICU583_ban.shl` script to do so.
14. Obtain the latest template makefiles from `$BANNER_HOME/general/misc` to `$BANNER_HOME/general/c`.
15. See C Compiling during Banner installation on page 139 to compile your Banner General C processes.

Related concepts

Managing Job Submission on UNIX on page 105

ICU upgrade for Windows-based environments

Upgrading to ICU version 58.3 on a Windows-based Job Submission server environment requires upgrading ICU to version 58.3 and upgrading Banner General C for ICU 58.3.

Download and upgrade ICU to version 58.3

Download the `icu4c-58_3-src.zip` file and upgrade a previous version of ICU to ICU version 58.3.

Before you begin

The ICU 58.3 upgrade requires Banner General 8.12 or later installed to ensure the appropriate objects exist in the `general/c` and `/general/misc` directories.

Files delivered with Banner General 8.12 in the `/general/c` directory:

- Makefile_tm_583
- tmcilib.cpp_583
- tmcilib.cpp_hpux_583
- tmcilib.h_583
- umsgtm.cpp_583
- umsgtm.h_583

Files delivered with Banner General 8.12 in the `/general/misc` directory:

- copyICU583_ban.bat
- copyICU583_ban.shl

- sctproc.pl_583

Procedure

1. Download the `icu4c-58_3-src.zip` file from Ellucian Download Center to a temporary location and review the `Readme.txt` file.
2. Log into the jobsub server as Administrator.
3. Stop the gurjobs process.
4. Rename the existing `icu` directory to `icu_`*`version`*, where *version* is the current version of ICU you are upgrading from.

   ```
   rename icu icu_36
   ```
5. Enter `mkdir icu` to create a new `icu` directory
6. Verify the following environment variables for *ICU_HOME* and *BANNER_HOME* exist and reference the appropriate directories. Establish these variables if they do not already exist.

   C:\>`echo %ICU_HOME%`

   C:\>`echo %BANNER_HOME%`

   The commands above return `E:\icu` and `E:\ellucian\banner\SMPL` respectively.
7. Change the directory to the *ICU_HOME* directory.
8. Copy the ICU 58.3 download file from the temporary location to the *ICU_HOME* directory.
9. Install ICU 58.3.
   a) Make sure you are logged in as Administrator and you are in the *ICU_HOME* directory.
   b) Unzip the downloaded `icu4c-58_3-src.zip` file to a directory one level above *ICU_HOME*.
      Unzipping the file creates files in the `icu` directory, so you need to unzip to the higher level directory.

      Note: The command example below shows unzipping the file to `E:\`, not `E:\icu`.

      E:\icu>`unzip -a icu4c-58_3-src.zip -d E:\`

      This process unzips approximately 4,000 files.

Related concepts

Managing Job Submission on Windows on page 103

Build ICU 58.3

Use Microsoft™ Visual Studio to compile the ICU 58.3 files.

Before you begin

Make sure you are using Visual Studio 2015.

Procedure

1. Open the `<`*`ICU`*`>\source\allinone\allinone.sln` workspace file in Visual Studio.

2. Select **Build** > **Configuration Manager**.
3. On the **Configuration manager** window, set the **Active solution configuration** field to Release and the **Active solution platform** field to Win32 or x64 depending on your operating system.
4. Select **View** > **Solution Explorer**.
5. Select **Build** > **Rebuild Solution**.

   If running the rebuild more than one time, select **Clean Solution** before you rebuild the solution.

   After the build completes a message displays indicating the status of the build similar to the following message:

   ```
   ======= Build: 29 succeeded, 0 failed, 0 up-to-date, 0 skipped =======
   ```
6. Run the following Visual Studio tests to validate ICU.

   a) In the **Solution Explorer** window, right-click **intltest** and select **Set as StartUp Project**.
   b) Press **Ctrl** + **F5** to run the test.
   c) Right-click **cintltst** and select **Set as StartUp Project**
   d) Press **Ctrl** + **F5** to run the test.
   e) Right-click **iotest** and select **Set as StartUp Project**
   f) Press **Ctrl** + **F5** to run the test.

   You might receive errors due to Banner modifications made to native ICU objects.

Update and compile Banner C objects

Update the Banner objects in the `%BANNER_HOME%/general/c` directory and compile all Banner C objects

About this task

After renaming obsolete Banner objects and copying updated objects in the `%BANNER_HOME%\general\c` directory, compile all Banner C objects using the delivered compile scripts for each product.

Procedure

1. Log into the jobsub server as Administrator.
2. Change the directory to `%BANNER_HOME%\general\misc`.

3. Enter `copyICU583_ban.bat`.

   This command renames the following two obsolete Banner objects with the name *`file_name`*`.obsolete.YYYMMDD_HHMI`:

   - `%BANNER_HOME%\general\c\msgfmttm.h`
   - `%BANNER_HOME%\general\c\msgfmttm.cpp`

   The command also copies the following new `_583` versions without the `_583` extension.

   - `%BANNER_HOME%\general\c\umsgtm.h`
   - `%BANNER_HOME%\general\c\umsgtm.cpp`
   - `%BANNER_HOME%\general\c\tmcilib.h`
   - `%BANNER_HOME%\general\c\tmcilib.cpp`
   - `%BANNER_HOME%\general\c\Makefile_tm`
   - `%BANNER_HOME%\general\misc\stproc.pl`
4. See C Compiling during Banner installation on page 139 to compile your Banner General C processes.
   a) Verify that the C objects compiled without errors by looking for "error" in the logs.
   b) Enter `dir %BANNER_HOME\general\exe\g* /o-d` to check for any objects with compile dates older than the current date.

      Note: There may be obsolete objects with an older compile date. These objects usually have a year designation as the last two characters.
5. After successfully compiling all C objects, start the gurjobs process and test your C processes.

Compiling C under UNIX

Compiling C code under the UNIX operating system is accomplished through the use of the make command, a special-purpose scripting language usually provided as part of the UNIX language development environment.

A makefile is needed for all but the most basic make operations; it specifies the actions to be taken to perform particular tasks, such as making an executable from a C source file, or building an object file from a Pro*C file.

To compile Banner Pro*C code into executables, you must create a makefile that includes all of the proper options and libraries for the combination of operating system, Oracle and compiler versions installed on your machine.

Related information

Article 000005986 Banner Example Makefiles

Using sctproc.mk

To use the sctproc.mk makefile, go to the directory containing the source code to be compiled, and enter a make command specifying both the makefile and the file to be generated.

Example:

```
make -f $BANNER_HOME/general/c/sctproc.mk gurtabl
```

Refer to your operating system documentation for further details on makefile construction and usage.

Added switch for `sctproc.mk` file

With Release 7.2, a manual change must be made to your site-specific `sctproc.mk` file. This change is necessary whether your site is using Oracle 9i or 10g.

Under Oracle 10g, 10.1.0.2, and 10.1.0.3, there is an Oracle issue that causes the Pro*C precompile to not recognize nested SQL INCLUDE statements. The `guaorac.c` file, used by every Banner Pro*C program, was modified for Release 7.2 to use the standard precompiler `#include` directive to include the `oraca.h` and `sqlca.h` files as a workaround for the defect. Because of this change, the manual change to `sctproc.mk` is necessary. In `sctproc.mk` you must specify an additional `–I` switch for the CFLAGS macro to include the `$ORACLE_HOME/precomp/public` directory. For example:

```
CFLAGS=-I. \
        -I$(GINC) \
        -I$(ORACLE_HOME)/precomp/public \
        -O $(ANSI) $(STRIP) $(CCHECK) $(ENV) \
        $(SCT_DEBUG) $(OTHER_C_FLAGS)
```

Reducing executable sizes

Pro*C executables may be extremely large on some UNIX platforms, with implications for both runtime performance and storage requirements. To reduce the size of executables, you may be able to use shared objects to provide dynamic linking at runtime. Refer to your Oracle and operating system documentation for further information.

If your operating system and Oracle release support dynamic linking, also known as shared objects, then modify your copy of sctproc.mk and add -lclntsh at the beginning of the LLIBS macro definition.

Example:

```
LLIBS=-lclntsh $(PROLDLIBS)
```

Also, the environment variable `LD_LIBRARY_PATH` will need to be defined in order for the shared object references to be resolved at runtime.

Example:

```
LD_LIBRARY_PATH=/u02/app/oracle/product/9.2.0/libexport LD_LIBRARY_PATH
```

Stripping your executables of debugging information may also significantly decrease their size; this can usually be done at compile time with the -s switch to the compiler, or later with the stand-alone strip program. Also, if you are using certain versions of SVR4-based operating systems (such as Dynix/ptx), the mcs-d command can be used to strip internal comments from the executables.

C Compiling during Banner installation

For an initial installation of Banner, all products that have C programs need to have them compiled. The Banner installation process uses the script bancc.shl on UNIX to compile all C code.

You may use the .shl (UNIX) or .pl (Windows) versions of the scripts below to compile a single product's C code. This step may execute for several hours depending on your machine speed and how many Banner products you are installing. The scripts are located in the appropriate product's / com or /misc production subdirectory.

Banner C compile procedures

The following is a list of the Banner products and their C compile procedures.

Banner Product	Compile Procedures
A/R System	tascmplc
Advancement	alucmplc
Finance	fincmplc
Financial Aid	rescmplc
General	gencmplc
Payroll	paycmplc
Position Control	poscmplc
Student	stucmplc

The executable files are built in the `$BANNER_HOME/general/exe` directory. This directory must be in the PATH of each Banner user.

UNIX

Below is a sample of the Banner General C compile shell script.

```
:# gencmplc.shl
#
```

```
cd $BANNER_LINKS
make -f $BANNER_HOME/general/c/sctproc.mk genobjs \
CHECKOPT=”sqlcheck=full userid=baninst1/u_pick_it”
make -f $BANNER_HOME/general/c/sctproc.mk gjrrpts
make -f $BANNER_HOME/general/c/sctproc.mk gurpded
make -f $BANNER_HOME/general/c/sctproc.mk glrletr
make -f $BANNER_HOME/general/c/sctproc.mk gppaddr
make -f $BANNER_HOME/general/c/sctproc.mk gurhelp
make -f $BANNER_HOME/general/c/sctproc.mk gurtabl
make -f $BANNER_HOME/general/c/sctproc.mk gurinso
make -f $BANNER_HOME/general/c/sctproc.mk gurskel
make -f $BANNER_HOME/general/c/sctproc.mk guaprpf
make -f $BANNER_HOME/general/c/sctproc.mk gurjobs \
CHECKOPT=”sqlcheck=full userid=baninst1/u_pick_it”
```

Windows

Below is a sample of the Banner General COBOL compile command script.

```
use sctcomp;
$sctcomp_product_dir = “general”;
$sctcomp_input_file_ref = \*DATA;
$sctcomp_c_process;
_END_
guastdf.c -exetype=obj -checkopt=full
guaorac2.pc -exetype=obj -checkopt=full
guawslp.c -exetype=obj
guarpfe.c -exetype=obj -checkopt=full
gjpprun.pc -checkopt=full
gjrrpts.pc
gurpded.pc
glrletr.pc
gppaddr.pc
gurhelp.pc
gurinso.pc -ckeckopt=full
gurskel.pc
gurtabl.pc
guaprpf.c
gurjobs.pc -checkopt=full
```

System-required data

Banner is a complex system with many parts that work together to manage your institution's data and to interact with users. When any one of the components of the system is missing, some of the system's functions may fail or may not work as intended.

In some cases, data itself can be considered an essential component of the system. The complete contents of certain tables, and specific rows in other tables, must be present for the system to work correctly. This essential data is called *system-required data*. System-required data is a subset of the seed data delivered with a new Banner installation. New Banner software releases often include seed data scripts that deliver additional system-required data.

Generally, Banner pages and processes will prevent you from deleting system-required data. But when you are using database tools or scripts to delete rows from the database—for example, during database cleanup to remove sample data before migrating into production—there is nothing to prevent essential data from being accidentally deleted. In those situations, you should take care not to delete any system-required data.

In many tables there is a **System-Required Indicator** column (for example, `SYSTEM_REQ_IND`). If the indicator has a value of `Y`, the row is presumed to be system required. But the system-required indicator is not a foolproof guide to Banner's system-required data, because:

- Some tables do not have a system-required indicator, but nonetheless contain essential data.
- Some tools and processes allow users to mark rows as system required, even if they are not essential for system operation.

This chapter lists system-required data for Banner General. Other system-required data is listed in the following documents:

- *Banner GTVSDAX Handbook*
- *Banner Accounts Payable TRM Supplement*
- *Banner Advancement TRM Supplement*
- *Banner Finance TRM Supplement*
- *Banner Financial Aid TRM Supplement*
- *Banner Human Resources TRM Supplement*
- *Banner Student TRM Supplement*

System-Required Tables

Tables owned by BANSECR

Tables owned by the BANSECR user ID provide the data for Banner's object/user security system, including the permissions that allow the components of the Banner system to operate.

These tables should never be included in automated database purge processes. If it becomes necessary to clean up the tables owned by BANSECR, it should be done carefully and manually by an administrator familiar with the institution's security setup.

Large tables

Some General tables are delivered with hundreds of system-required rows, and it would be impractical to reprint their complete data here. Before making changes to these tables, you may want to save an export of their data in case it becomes necessary to restore them later.

- Report/Process Definition Table (GJBJOBS)
- Jobs Parameter Definition Table (GJBPDEF)
- Default Parameter Table (GJBPDFT)
- General Jobs Parameter Value Table (GJBPVAL)
- Letter Generation Variable Base Table (GLBVRBL)
- Population Selection Rules Table (GLRSLCT)
- Letter Generation Variable Select Table (GLRVFRM)
- Letter Generation Variable Rules Table (GLRVRBL)
- Third-Party Function Calls Table (GOBFNXR)
- Parameter Group Code Rule Table (GOREQPG)
- Third-Party Function Parameters Table (GORPPRM)
- Third-Party Electronic Controls Table (GORTCTL)
- Voice Response Controls Table (GORVCTL)
- Parameters Table (GORWFPM)
- EDI Standard Code Validation Table (GTVSCOD)
- SEVIS Consular Post Codes Validation Table (GTVSVCP)
- Banner Module URL table (GUBMODU)
- Banner Page Module Mapping table (GUBPAGE)
- General Menu Repeating Table (GURMENU)
- Banner Business Entity Table (GURMESG)
- Option Menu Repeating Table (GUROPTM)

Other tables

The Crosswalk Validation Table (GTVSDAX) contains important delivered data. See the *Banner GTVSDAX Handbook* for complete details.

Seed data for the FGAC Domain Policy Table (GORFDPL) is documented in the *Banner Data Security Handbook* (formerly titled *Banner FGAC Handbook*).

The Institutional Description Table (GUBINST) must contain at least one row. It is delivered with example data that you can modify or replace with your own institution’s data.

Normally, you will have no reason to edit the following tables that contain system-required data:

- The General Version Tracking Table (GURVERS), which tracks the version history of the Banner General product
- The Banner 9 Database version history Table (GURWADB), which tracks the history of the Banner 9 database products.
- The Banner 9 Application version history Table (GURWAPP), which tracks the history of the Banner 9 application products.

System-Required Rows

Specific delivered, system-required values are listed in this section, organized alphabetically by table name.

Even though these are considered system-required values, not all of the values listed here need to be present in every institution's Banner database. Many of the tables and values support specific Banner systems, subsystems, and functions. If your institution does not use those components of Banner, then the absence of the corresponding data will not cause any problem.

As an example, the FGAC Domain Driver Table Table (GOBFDMN) maintains driver tables for VBS and PII processing. In the section below, you will see driver tables listed in GOBFDMN for all of the Banner products. If your institution has not implemented Banner Finance, for example, then the absence of Finance driver table entries in the GOBFDMN table would not cause any problems.

GLBAPPL - Letter Generation Application Table

Application	Description	System Code
ALUMNI	BANNER Alumni/ Development	A
FINAID	Financial Aid Application	R
WKBOOK	Sample Application for G01C	G
COURTS	Banner Courts	C
HRAPPL	HR Applicant	H
HREMPL	HR Employee	H

GLBAPPL - Letter Generation Application Table		
Application	**Description**	**System Code**
WORKBOOK	Letter Generation Workbook	S
STUDENT	Student Module	S
GENERAL	General Module	G
RECRUITING	BANNER Student Recruiting Mod.	S
ADMISSIONS	BANNER Student Admissions Mod.	S
HOUSING	BANNER Student Housing Module	S
BANSTU_SAMPLE	Student Sample Data Examples	S
PIN_RESET	PIN Reset Notification	G
COBRA_APPL	Cobra Application	H

GLBOBJT	**Letter Generation Object Base Table**	
	MARRIED	Select Married Persons
	DIVORCED	Select Divorced Persons
	SINGLE	Select Single Persons
	NOT_DEAD	Not Dead Rules
	WOMEN	Select Women
	RECR_TERM	Recruit Term
	MEN	Select Men
	RCRAPP1-RCRAPP2_JOINS	table joins
	RECR_COLL	Recruit College
	RECR_MAJR	Recruit Major
	RECR_LEVL	Recruit Level
	PERSON_RECORD	person
	CURRENT_NAME_ID	most accurate name and ID

GLBSLCT - Population Selection Base Table			
Application	**Selection**	**Creator ID**	**Description**
FINAID	NOVER_NOTPACKAGED	FAISMGR	not selected - not packaged

GLBSLCT - Population Selection Base Table			
Application	**Selection**	**Creator ID**	**Description**
FINAID	TEMP	FAISUSR	Temporary
STUDENT	199510_NEW_FROSH	SAISUSR	199510 New Frosh Registrations
STUDENT	199510_NEW_UG_FROSH	SAISUSR	199510 New Frosh Enrollees
ALUMNI	CLASS72	ADISUSR	Alumni by preferred class
ALUMNI	CLASS86	ADISUSR	Alumni by preferred class
ALUMNI	PREF_CLASS	ADISUSR	Alumni by preferred class
ALUMNI	PROSPECTS	ADISUSR	All prospects
ALUMNI	PROS_RESEARCH	ADISUSR	Prospect Research Population
FINAID	PRIORITY_LATE	FAISMGR	late applicants
FINAID	PRIORITY_ONTIME	FAISMGR	on time applicants
RECRUITING	199301_RECRUITS	SAISUSR	Selection of 1993 Recruits
FINAID	SELECTED-NOTCOMP	FAISMGR	sel for verif. - not completed
FINAID	DORM	FAISMGR	Housing Code Selection
FINAID	UMETNEED	FAISMGR	Need
FINAID	UNMET	FAISMGR	Need
FINAID	MANUAL	FAISMGR	manual pop selection
FINAID	ALL_REQ_COMP	FAISUSR	All Requirements Complete
FINAID	AWARD_LTR	FAISPRD	Students Needing Award Letters
FINAID	NEEDY_FROM_PA	FAISUSR	Needy From PA
FINAID	NEEDY	FAISPRD	Students With Large Gross Need
HREMPL	DEDN	HRISUSR	Employee Dedn List
ALUMNI	NEGATIVE_AMOUNT_DUE	ADISUSR	Negative amount due-membership

GLBSLCT - Population Selection Base Table			
Application	**Selection**	**Creator ID**	**Description**
ALUMNI	FORM_NOT_RECEIVED	ADISUSR	Matching Gift Form Not Recvd
ALUMNI	GROUPED_GIFTS_IDS	ADISUSR	IDs with grouped gifts
WKBOOK	MEN	SAISUSR	Select All Men
FINAID	TRACK2	FAISMGR	Never had a tracking letter
FINAID	TRACK1	FAISMGR	Track Letter not sent
FINAID	AWARD_FLAG	FAISMGR	Award Letter Flag = ‘Y’
FINAID	TRACK_FLAG	FAISMGR	Track Letter Flag = ‘Y’
FINAID	VA_BENEFITS	FAISMGR	receiving va benefits
FINAID	SS_BENEFITS	FAISMGR	Receiving SS benefits
STUDENT	199510_NEW_UG_TRAN	SAISUSR	199510 New Frosh Enrollees
FINAID	CHILD_CARE	FAISMGR	have dependent child expenses
FINAID	NONCITIZEN	FAISMGR	not a U.S. citizen
BANSTU_SAMPLE	TS_CONTRACTS	SAISUSR	Students with Contracts
BANSTU_SAMPLE	TS_EXEMPTIONS	SAISUSR	Students with Exemptions
FINAID	MANUAL	FAISUSR	Manual
BANSTU_SAMPLE	199610_ENROLLED	SAISUSR	199610 enrolled students
BANSTU_SAMPLE	MEN	SAISUSR	Select Men
WKBOOK	199610_ENROLLED	SAISUSR	199610 Enrolled Students
BANSTU_SAMPLE	199510_NEW_AND_TRANS	SAISUSR	199510 N & T enrolled UG
WKBOOK	199610_CURR_STU	SAISUSR	199610 Current Students
FINAID	RORSTAT_RECORD	FAISMGR	has rorstat record in aid year
BANSTU_SAMPLE	199510_UG_NEW	SAISUSR	199510 Ug, New
ADMISSIONS	199610_APPLICANTS	SAISUSR	Fall 1996 Applicants
FINAID	AFDC	FAISMGR	afdc recipient

GLBSLCT - Population Selection Base Table			
Application	**Selection**	**Creator ID**	**Description**
FINAID	RCRAPP_RECORD	FAISMGR	Need Analysis Records
FINAID	BGRP	FAISMGR	all students in budget group
FINAID	TGRP	FAISMGR	all students in track group
FINAID	PGRP	FAISMGR	all students in pckg group
FINAID	CHILDSUPPORT	FAISMGR	receive child support
FINAID	CITIZENSHIP	FAISMGR	citizenship verification
FINAID	COMPLETE_DISB_REQ	FAISMGR	all disb req. complete
FINAID	COMPLETE_PCKG_REQ	FAISMGR	all pckg req. complete
FINAID	COMPLETE_TRACKING	FAISMGR	all requirements complete
HOUSING	HOUSING_ASSIGNMENTS	SAISUSR	Active Housing Assignments
FINAID	MANUAL1	FAISUSR	manual1
For all entries listed above, **Lock Indicator** is N, and **Type** is null.			

GLRAPPL Letter Generation Application Rules Table								
Application	**Seq. No.**	**Line No.**	**Data Element**	**Operator**	**Value**	**(**	**)**	**AND/OR**
ADMISSIONS	5	1	SARADAP_TERM_CODE_ENTRY	=	&APPLICATION_TERM			
RECRUITING	4	1	SRBRECR_TERM_CODE	=	&RECRUITING_TERM			
COURTS	1	1	CDBCASE_ID	IS NOT NULL				
HRAPPL	1	1	PABAPPL_PIDM	=	SPRIDEN_PIDM			AND
HRAPPL	2	2	SPRIDEN_ENTITY_IND	=	P			AND
HRAPPL	3	3	SPRIDEN_CHANGE_IND	IS NULL				
HREMPL	1	1	PEBEMPL_PIDM	=	SPRIDEN_PIDM			AND

GLRAPPL	Letter Generation Application Rules Table							
Application	**Seq. No.**	**Line No.**	**Data Element**	**Operator**	**Value**	**(**	**)**	**AND/OR**
HREMPL	2	2	SPRIDEN_ENTITY_IND	=	P			AND
HREMPL	3	3	SPRIDEN_CHANGE_IND	IS NULL				
WORKBOOK	1	1	SPRIDEN_CHANGE_IND	IS NULL				AND
WORKBOOK	2	2	SPRIDEN_ENTITY_IND	=	P			AND
WORKBOOK	3	3	SPBPERS_DEAD_IND	IS NULL				

GLROBJT	Letter Generation Object Rules Table							
Object	**Seq. No.**	**Line No.**	**Data Element**	**Operator**	**Value**	**(**	**)**	**AND/OR**
RECR_COLL	1	1	SRBRECR_COLL_CODE	=	&recr_coll			
MARRIED	1	1	SPBPERS_MRTL_CODE	=	M			
DIVORCED	1	1	SPBPERS_MRTL_CODE	=	D			
SINGLE	1	1	SPBPERS_MRTL_CODE	=	S			
NOT_DEAD	1	1	SPBPERS_DEAD_IND	IS NULL				
WOMEN	1	1	SPBPERS_SEX	=	F			
RECR_COLL	1	1	SRBRECR_COLL_CODE	=	&recr_coll			
MARRIED	1	1	SPBPERS_MRTL_CODE	=	M			
DIVORCED	1	1	SPBPERS_MRTL_CODE	=	D			
SINGLE	1	1	SPBPERS_MRTL_CODE	=	S			
NOT_DEAD	1	1	SPBPERS_DEAD_IND	IS NULL				
WOMEN	1	1	SPBPERS_SEX	=	F			

GLROBJT Letter Generation Object Rules Table

Object	Seq. No.	Line No.	Data Element	Operator	Value	(	)	AND/OR
RECR_TERM	1	1	SRBRECR_TERM_CODE	=	&recr_term			
MEN	1	1	SPBPERS_SEX	=	M			
RCRAPP1-RCRAPP2_JOINS	1	1	RCRAPP1_PIDM	=	RCRAPP2_PIDM			AND
RCRAPP1-RCRAPP2_JOINS	2	2	RCRAPP1_AIDY_CODE	=	RCRAPP2_AIDY_CODE			AND
RCRAPP1-RCRAPP2_JOINS	3	3	RCRAPP1_INFC_CODE	=	RCRAPP2_INFC_CODE			AND
RCRAPP1-RCRAPP2_JOINS	4	4	RCRAPP1_SEQ_NO	=	RCRAPP2_SEQ_NO			
RECR_MAJR	1	1	SRBRECR_MAJR_CODE	=	&recr_majr			
RECR_LEVL	1	1	SRBRECR_LEVL_CODE	=	&recr_levl			
PERSON_RECORD	1	1	SPRIDEN_ENTITY_IND	=	P			
CURRENT_NAME_ID	1	1	SPRIDEN_CHANGE_IND	IS NULL				

GLRSFRM - Population Selection Select Table

Application	Selection Code	Select Clause	From Clause	Order By	Group By
ALUMNI	CLASS72	APBCONS_PIDM	APBCONS		
ALUMNI	CLASS86	APBCONS_PIDM	APBCONS		
ALUMNI	PREF_CLASS	APBCONS_PIDM	APBCONS		
ALUMNI	PROSPECTS	AMRINFO_PIDM	AMRINFO		
ALUMNI	PROS_RESEARCH	AMRPUSR_PIDM	AMRPUSR, APBCONS		

GLRSFRM - Population Selection Select Table

Application	Selection Code	Select Clause	From Clause	Order By	Group By
RECRUITING	199301_ RECRUITS	SRBRECR_ PIDM	SRBRECR		
FINAID	DORM	RORSTAT_PIDM	RORSTAT	RCRAPP1	
FINAID	UMETNEED	RORSTAT_PIDM	RORSTAT	RCRAPP1	
FINAID	UNMET	RORSTAT_PIDM	RORSTAT	RCRAPP1	
HREMPL	DEDN	PDRDEDN_PIDM	PDRDEDN		
FINAID	ALL_REQ_COMP	RORSTAT_PIDM	RORSTAT		
FINAID	AWARD_LTR	RORSTAT_PIDM	RORSTAT		
FINAID	NEEDY	RORSTAT_PIDM	RORSTAT		
FINAID	NEEDY_FROM_ PA	RORSTAT_PIDM	RORSTAT	RCRAPP1	
FINAID	AFDC	RORSTAT_PIDM	RORSTAT, RCRAPP1		
FINAID	BGRP	RORSTAT_PIDM	RORSTAT		
FINAID	TGRP	RORSTAT_PIDM	RORSTAT		
ALUMNI	NEGATIVE_ AMOUNT_DUE	AARMEMB_PIDM	AARMEMB A		
FINAID	PGRP	RORSTAT_PIDM	RORSTAT		
FINAID	CHILDSUPPORT	RORSTAT_PIDM	RORSTAT, RCRAPP1		
FINAID	CITIZENSHIP	RORSTAT_PIDM	RORSTAT, RCRAPP1		
FINAID	COMPLETE_ DISB_REQ	RORSTAT_PIDM	RORSTAT		
FINAID	COMPLETE_ PCKG_REQ	RORSTAT_PIDM	RORSTAT		
FINAID	COMPLETE_ TRACKING	RORSTAT_PIDM	RORSTAT		
FINAID	NOVER_ NOTPACKAGED	RORSTAT_PIDM	RORSTAT, RCRAPP1		
FINAID	PRIORITY_ LATE	RORSTAT_PIDM	RORSTAT		
FINAID	PRIORITY_ ONTIME	RORSTAT_PIDM	RORSTAT		

GLRSFRM - Population Selection Select Table					
Application	**Selection Code**	**Select Clause**	**From Clause**	**Order By**	**Group By**
FINAID	SELECTED-NOTCOMP	RORSTAT_PIDM	RORSTAT, RCRAPP1		
ALUMNI	FORM_NOT_RECEIVED	AGBMGID_EMPL_PIDM	AGBGIFT, AGBMGID		
ALUMNI	GROUPED_GIFTS_IDS	AGRRCPT_PIDM	AGRRCPT		
WKBOOK	MEN	SPBPERS_PIDM	SPBPERS		
FINAID	TRACK2	RORSTAT_PIDM	RORSTAT, RRRAREQ		
FINAID	TRACK1	RORSTAT_PIDM	RORSTAT	RRRAREQ	SPRIDEN GURMAIL A
FINAID	AWARD_FLAG	RORSTAT_PIDM	RORSTAT	SPRIDEN	
FINAID	TRACK_FLAG	RORSTAT_PIDM	RORSTAT	SPRIDEN	
FINAID	VA_BENEFITS	RORSTAT_PIDM	RORSTAT, RCRAPP1		
FINAID	SS_BENEFITS	RORSTAT_PIDM	RORSTAT	RCRAPP1	
STUDENT	199510_NEW_FROSH	SGBSTDN_PIDM	SGBSTDN A		
STUDENT	199510_NEW_UG_FROSH	SGBSTDN_PIDM	SGBSTDN A		
STUDENT	199510_NEW_UG_TRAN	SGBSTDN_PIDM	SGBSTDN A		
FINAID	CHILD_CARE	RCRAPP3_PIDM	RCRAPP1	RCRAPP3	
FINAID	NONCITIZEN	RCRAPP1_PIDM	RCRAPP1		
BANSTU_SAMPLE	TS_EXEMPTIONS	TBBESTU_PIDM	TBBESTU		
BANSTU_SAMPLE	199610_ENROLLED	SFBETRM_PIDM	SFBETRM		
BANSTU_SAMPLE	MEN	SPBPERS_PIDM	SPBPERS		
WKBOOK	199610_ENROLLED	SFBETRM_PIDM	SFBETRM		
BANSTU_SAMPLE	199510_NEW_AND_TRANS	SGBSTDN_PIDM	SGBSTDN A		
WKBOOK	199610_CURR_STU	SGBSTDN_PIDM	SGBSTDN A		

GLRSFRM - Population Selection Select Table					
Application	**Selection Code**	**Select Clause**	**From Clause**	**Order By**	**Group By**
FINAID	RORSTAT_ RECORD	RORSTAT_PIDM	RORSTAT	SPRIDEN	
BANSTU_ SAMPLE	199510_UG_ NEW	SGBSTDN_PIDM	SGBSTDN A		
ADMISSIONS	199610_ APPLICANTS	SARADAP_PIDM	SARADAP		
FINAID	RCRAPP_ RECORD	RORSTAT_PIDM	RCRAPP1	RORSTAT	
HOUSING	HOUSING_ ASSIGNMENTS	SLRRASG_PIDM	SLRRASG, STVASCD		

GOBDIRO	Directory Options Rule Table		
Directory Code	**Directory Type**	**Item Type**	**Sequence Number**
NAME	A	N	1
ADDR_PR	A	A	2
TELE_PR	A	T	3
ADDR_CP	S	A	4
TELE_CP	S	T	5
ADDR_OF	E	A	6
TELE_OF	E	T	7
TELE_FAX	A	T	8
DEPT	E	N	9
GRD_YEAR	S	N	10
COLLEGE	S	N	11
TITLE	E	N	12
EMAIL	A	N	13
MAIDEN	D	N	14
ADDR_HO	A	A	15
TELE_HO	A	T	16
ADDR_BU	A	A	17
TELE_BU	A	T	18
CLASS_YR	D	N	19

GOBDIRO	**Directory Options Rule Table**		
Directory Code	**Directory Type**	**Item Type**	**Sequence Number**
`PR_COLL`	D	N	20
For all entries above, **Included in Directory** is `N`, **Display in Directory** is `N`, and **Default Indicator** is `N`.			

GOBFDMN - FGAC Domain Driver Table Table			
Domain Code	**Table Name**	**Type**	**PII Column Name**
`TB_ACCOUNT_PII`	TBRACCD	PII	`TBRACCD_PIDM`
`FB_CUSTOMER_PII`	FTVCUST	PII	`FTVCUST_PIDM`
`FB_EMPLOYEE_PII`	FCBEMPL	PII	`FCBEMPL_PIDM`
`FB_VENDOR_PII`	FTVVEND	PII	`FTVVEND_PIDM`
`FB_MANAGER_PII`	FTVFMGR	PII	`FTVFMGR_FMGR_CODE_PIDM`
`FB_AGENCY_PII`	FTVAGCY	PII	`FTVAGCY_AGCY_CODE_PIDM`
`GB_FGACACCESS_VBS`	GOBFGAC	VBS	
`GB_FGAC_PREDICATE_VBS`	GORFPRD	VBS	
`GB_INTERNATIONAL_VBS`	GOBINTL	VBS	
`GB_SPRADDR_VBS`	SPRADDR	VBS	
`GB_SPRMEDI_VBS`	SPRMEDI	VBS	
`GB_SPRTELE_VBS`	SPRTELE	VBS	
`PB_APPLICANT_PII`	PABAPPL	PII	`PABAPPL_PIDM`
`PB_BENEFITS_PII`	PDRBENE	PII	`PDRBENE_PIDM`
`PB_COBRA_PII`	PCBPERS	PII	`PCBPERS_PIDM`
`RB_FINAID_PII`	RORSTAT	PII	`RORSTAT_PIDM`
`SB_ADMISSIONS_PII`	SARADAP	PII	`SARADAP_PIDM`
`SB_FACULTY_PII`	SIBINST	PII	`SIBINST_PIDM`
`SB_HOUSING_PII`	SLBRMAP	PII	`SLBRMAP_PIDM`
`SB_GENSTUDENT_PII`	SGBSTDN	PII	`SGBSTDN_PIDM`
`SB_RECRUIT_PII`	SRBRECR	PII	`SRBRECR_PIDM`
`SB_REGISTRATION_PII`	SFBETRM	PII	`SFBETRM_PIDM`
`SB_TRANSFER_PII`	SHRTTRM	PII	`SHRTTRM_PIDM`
`AB_CONSTITUENT_PII`	APBCONS	PII	`APBCONS_PIDM`

GOBFDMN - FGAC Domain Driver Table Table			
Domain Code	**Table Name**	**Type**	**PII Column Name**
`AB_ORG_PII`	AOBORGN	PII	`AOBORGN_PIDM`
`SB_RECRUIT_VBS`	SRBRECR	VBS	
`PB_EMPLOYMENT_PII`	PEBEMPL	PII	`PEBEMPL_PIDM`
`SB_LEARNER_VBS`	SGBSTDN	VBS	
`SB_CATALOG_VBS`	SCBCRSE	VBS	
`SB_SCHEDULE_VBS`	SSBSECT	VBS	
`RB_FINAID_VBS`	RORSTAT	VBS	
`RB_FINAID_STUDENT_VBS`	RORSTAT	VBS	
`SB_CURRICULUM_VBS`	SORLCUR	VBS	
`SB_FIELDOFSTUDY_VBS`	SORLFOS	VBS	
`SB_ADMISSIONS_VBS`	SARADAP	VBS	
`SB_TESTCODES_VBS`	STVTESC	VBS	
`SB_TESTSCORE_VBS`	SORTEST	VBS	
`SB_OTHERGPA_CODES_VBS`	STVGPAT	VBS	
`SB_OTHERGPA_VBS`	SORGPAT	VBS	
`SB_OTHERGPA_STUDENT_VBS`	SORGPAT	VBS	
`SB_TESTSCORE_STUDENT_VBS`	SORTEST	VBS	
For all entries above, **Enable PII Indicator** is delivered with the value `N`.			

GOBFDTP - FGAC Domain Type Rule Table		
	Domain Type Code	**Predicate Indicator**
	PII	N
	VBS	Y

GOBFEOB

Objects Excluded from FGAC Processing Rules Table

This table lists objects that bypass FGAC rules. Objects listed in GOBFEOB have full access to data regardless of VBS or PII rules that might otherwise apply. Following is the complete list of objects included in seed data for GOBFEOB, organized alphabetically.

```
AAPACKN, AAPADJS, AAPCARD, AAPFEED, AAPREMD, AAPRNEW, AAPSTAT, ADPACCT,
 ADPCFAE, ADPEXPD, ADPFEED, ADPPFED, ADPVSER, AFPCAMR, AFPDONR,
 AFPSOLA, AFPSOLB, AFPSOLC, AFPTELF, AGPACCT, AGPACKN, AGPACKR,
 AGPADJS, AGPALMP, AGPCASH, AGPDCGL, AGPGANL, AGPGCOM, AGPLYSY,
 AGPMATA, AGPMATC, AGPMATF, AGPMATG, AGPMATS, AGPPACT, AGPPOUT,
 AGPREM1, AGPREM2, AGPSCTA, AGPTLMK, ALPMAIL, ALPMSEL, APAPPFL,
 APPAPFL, APPCEN1, APPCEN2, APPCLST, APPCONS, APPCUPD, APPDCAR,
 APPDCLB, APPDCLS, APPDEXT, APPDFLS, APPDPRC, APPSTDI, ASPSOLA,
 ASPSOLB, ASPSORL, AXPMATG
BWPREDIR
CRQC3000
FAB1099, FABCHK1, FABCHKA, FABCHKD, FABCHKP, FABCHKR, FABCHKS, FABMATC,
 FAM1099, FAPCARD, FAPCDIR, FAPDIRD, FAPINVT, FAPTREG, FARAAGE,
 FARBBAL, FARBREC, FARCHKR, FARCSHR, FARDIRD, FARIAGE, FARINVA,
 FARINVS, FARIREC, FAROINV, FARVALP, FARVHST, FARVNUM, FARWHLD,
 FARWHLY, FAT1099, FATCHKS, FBRAPPD, FBRAPPR, FBRBDBB, FBRBDDS,
 FBRBDRL, FBRFEED, FBRMCHG, FBRWKSH, FCBBILL, FCBEQPT, FCBINVT,
 FCBLABR, FCBMATL, FCRBDTR, FCRSCHD, FCRVARA, FEPOEXT, FFPDEPR,
 FFPOEXT, FFRAGRP, FFRDTGA, FFRDTGT, FFRMAST, FFRPROC, FFRPROP,
 FGPGEXT, FGRACCI, FGRACTG, FGRACTH, FGRACTV, FGRAGYH, FGRBAVL,
 FGRBDRL, FGRBDSC, FGRBIEX, FGRBLSH, FGRCASH, FGRCBSR, FGRCGBA,
 FGRCGBS, FGRCHFB, FGRCHNA, FGRCLOP, FGRCOBS, FGRCREF, FGRCSBA,
 FGRCSCF, FGRCSRE, FGRCSRP, FGRCSSR, FGRCTRL, FGRCUNA, FGRENRL,
 FGRFAAC, FGRFBAL, FGRFITD, FGRFNDH, FGRFPSN, FGRGLEX, FGRGLRL,
 FGRGLTA, FGRIDOC, FGRJVLR, FGRLOCH, FGRODTA, FGROPNE, FGRORGH,
 FGRPDTA, FGRPRAP, FGRPRAR, FGRPRGH, FGRREOB, FGRREOC, FGRTAXR,
 FGRTBAL, FGRTBEX, FGRTOFR, FGRTRNH, FGRTRNI, FGRTRNR, FIRBVAL,
 FIRDIST, FIRLINK, FIRPVAL, FIRRDST, FIRUNIT, FNPGAIN, FNPSPND,
 FNPUNTZ, FNRHIST, FNRPRNC, FNRSPNC, FOIIDEN, FORAPPL, FPABIDD,
 FPACORD, FPAPORD, FPARQST, FPPPOBC, FPRBEVL, FPRDELV, FPROPNP,
 FPROPNR, FPRPURA, FPRRCDL, FPRRCST, FPRVCAT, FPRVVOL, FPTBIDD,
 FPTPORD, FPTRQST, FRPBINF, FRPGINF, FRPMESG, FRR134B, FRR269R,
 FRR270B, FRR272B, FRR272R, FRRABUD, FRRAGES, FRRAGYH, FRRBDEX,
 FRRBEXC, FRRBILL, FRRBREV, FRRBUDG, FRRCNSF, FRREVNG, FRREVNP,
 FRRFEXC, FRRGBFY, FRRGENB, FRRGENR, FRRGITD, FRRGPFY, FRRGRNT,
 FRRGRPT, FRRGRTN, FRRINDC, FRRINVS, FRRTRNR, FSRDTLG, FSRINVL,
 FSRISST, FSRLWSR, FSROPNR, FSROUTP, FSRPHYR, FSRPICK, FSRPIDR,
 FSRPIWS, FSRPUTL, FSRSTEX, FSRSUPC, FUPLOAD, FURAPAY, FURFEED
GJRRPTS, GLBDATA, GLBLSEL, GLBPARM, GLOLETT, GLRLETR, GOAEACC, GOAFPII,
 GOAMTCH, GORPGEO, GORSEVE, GORSGEO, GPPADDR, GUASYST, GUAVRFY,
 GUIALTI, GUPDELT, GURDETL, GURHELP, GURINSO, GURPDED, GURTABL,
 GURTEXT, GURTPAC
HWPREDIR, HWSRCTLG, HWSRSCHD
IRRKAWD, IRRKTRK, IRRKTRN, ISRKADM, ISRKBIL, ISRKCRS, ISRKGRD, ISRKSCH,
 ISRKTRN
NBPBROL, NBPBUDM, NBPENCB, NBPMASS, NBPSPEX, NBPSPUP, NBRBWRK, NBRPCLS,
 NBRPINC, NBRPOSN, NBRPSTA, NHPFIN1, NHPFIN2, NHRBDST, NHRDIST,
 NHRECRT, NHREDST, NHRSDST, NOPEAMA, NORAPTR
PARAPPL, PARMAPP, PARREQS, PCRCORT, PCRLTRS, PCRNOTF, PCRRATE, PDP1042,
 PDPBDMC, PDPCFLX, PDPF496, PDPFLEX, PDPLIFE, PDPMR87, PDPPERS,
 PDRBCOV, PDRBFDN, PDRBLST, PDRFLEX, PDRFLXU, PDRFUPT, PDRLIFE,
 PEP1042, PEPAEXT, PEPCSAL, PEPEDEX, PEPFACL, PEPPCRE, PERAPND,
 PERCAF7, PEREO11, PEREO1D, PEREO41, PEREO4D, PEREO51, PEREO5D,
 PEREO61, PEREO62, PEREO6D, PERFACL, PERHIRE, PERLEAV, PERORGC,
 PEROSHA, PERPAPP, PERPGAN, PERPHIR, PERPTER, PERREVW, PERROEC,
 PERTERM, PERTTTT, PERUTAN, PERV100, PERWFAN, PHPBOND, PHPBREC,
 PHPCALC, PHPCDIR, PHPCHEK, PHPCHKL, PHPDIRD, PHPDOCM, PHPFEXP,
 PHPLEAV, PHPMTIM, PHPPROF, PHPRETO, PHPTIME, PHPUPDT, PHRCDST,
 PHRCISS, PHRCOST, PHRDCON, PHRDERR, PHRDIRD, PHRDREG, PHRDSTT,
 PHRFACE, PHRHOUR, PHRLGST, PHRLRAR, PHRORGT, PHRPREG, PHRROST,
 PHRSTCA, PHRTMSH, PHRTREG, PORADUT, PORAUDT, PORPPFL, PPRSINV,
 PXP1099, PXPMT42, PXPMTT4, PXPMTTA, PXPMTTN, PXPW2MM, PXPW2MP,
 PXPW2TP, PXR1042, PXR1099, PXRASCD, PXRLIST, PXRP941, PXRROEC,
 PXRT4AC, PXRT4AN, PXRT4CN, PXRTDEP, PXRW2PR, PXRW2US
RBRABUD, RBRBCMP, RCBCT05, RCBCT06, RCBTP05, RCBTP06, RCMATCH, RCPDTMP,
 RCPIMFM, RCPMTCH, RCRTP03, RCRTP04, RCRTP05, RCRTP06, REBCD00,
 REBCD01, REBCD02, REBCD03, REBCD04, REBCD05, REBCD06, RERCALX,
 RERCRCR, REREX03, REREX04, REREX05, RERFI00, RERFI01, RERFI02,
 RERFI03, RERFI04, RERFI05, RERFI06, RERIM03, RERIM04, RERIM05,
 RERIMEX, RERIS00, RERIS01, RERIS02, RERIS03, RERIS04, RERIS05,
 RERIS06, RERPELL, RERPL01, RERPL02, RERPL03, RERPL04, RERPL05,
 RERPR00, RERPR01, RERPR02, RERPR03, RERPR04, RERPR05, RESDTMP,
 RFRABAL, RFRBUDG, RFRFUND, RFRSBAL, RHRCOMM, RHRFATR, RHRTRAN,
 RJRAUTH, RJRDPPR, RJRLOAD, RJRPAYE, RJRSEEC, RLRLETR, RLRLOGG,
 RNEIN00, RNEIN01, RNEIN02, RNEIN03, RNEIN04, RNEIN05, RNEIN06,
 RNRPINI, RNRTMAC, RNRTMNE, RNRTMNI, RNRVRFY, ROBBGRP, ROESAPR,
 ROOAUTO, ROOGSQL, ROPROLL, ROPSAPR, RORALOG, RORAPLT, RORBPST,
 RORCALC, RORFS00, RORFS01, RORFS02, RORFS03, RORFS04, RORGRDE,
 RORREGS, RORUSER, RPBDDRV, RPBLMIA, RPBLMID, RPBLMIE, RPBPDRV,
 RPBVDRV, RPBVLDT, RPEDISB, RPEPCKG, RPEPELL, RPEPINT, RPRADSB,
 RPRAWDB, RPRAWRD, RPRCNCL, RPRCP01, RPRCP02, RPRCP03, RPRCP04,
 RPRCP05, RPRDDUP, RPRDLLC, RPRDLLR, RPRDLPM, RPRDSPT, RPRDU00,
 RPRDU01, RPRDU02, RPRDU03, RPRDU04, RPRDU05, RPREFTL, RPREFTP,
 RPRELAP, RPRELAX, RPRELCT, RPRELRU, RPRHDRL, RPRLNAG, RPRLNEX,
 RPRLODE, RPRLORC, RPRLORE, RPRLSUM, RPRPNPT, RPRRECD, RPRRECN,
 RPRSAWD, RPRSBPR, RPRSTCR, RPRTIVC, RPRTIVI, RPRTIVR, RPRVABN,
 RRRAREQ, RRREXIT, RRRTRAN, RSRDSCP, RSRENRL, RSRPCOL
SADA3202, SADA3203, SADA3204, SADA3205, SADA3206, SADAFLEX, SADAPRNT,
 SAPADMS, SAPAMAL, SAR189U, SARACTM, SARADMS, SARAMAL, SARAMAS,
 SARAMCV, SARAMDP, SARAMXF, SARBDSN, SARDCBT, SARDCSN, SAREMAL,
 SARETBL, SARETMT, SARETPG, SARRATE, SATAMCS, SCRBULT, SCROIMS,
 SCRRIMS, SCTC1500, SCTC2000, SCTC3000, SCTD0600, SCTH1000, SERADAL,
 SERCBREC, SERCCRC, SERLOAD, SERPSRC, SERSAREC, SERSBREC, SERSDREC,
 SERSEREC, SERSIREC, SERSMREC, SERSPREC, SERSVRC, SERSXREC, SERXBREC,
 SERXCREC, SERXEREC, SERXFREC, SFPAGRD, SFPBLCK, SFPCREQ, SFPENRL,
 SFPFAUD, SFPFREQ, SFPREGS, SFPWAIT, SFRENRL, SFRFASC, SFRFEES,
 SFRHCNT, SFRLINK, SFRNOWD, SFRNSLC, SFRPINI, SFRRGAM, SFRRNOP,
 SFRSCHD, SFRSLST, SFRSSCR, SFRTMST, SFRWDRL, SGPBLCK, SGPCOOP,
 SGPHOLD, SGPSTDN, SGRCHRT, SGRKNOW, SGRSTDN, SGRVETN, SHPTAEQ,
 SHPTRTC, SHRASTD, SHRCATT, SHRCGPA, SHRCIPC, SHRCOMM, SHRCONV,
 SHRDEGS, SHRDEGV, SHREDII, SHREDIP, SHREDIR, SHREDIY, SHRETRP,
 SHRGPAC, SHRGRDE, SHRIACT, SHRIAGE, SHRICIP, SHRIETH, SHRIGRS,
 SHRIPDS, SHRIQUS, SHRIRES, SHRPREV, SHRROLL, SHRRPTS, SHRTAEQ,
 SHRTECA, SHRTPOP, SHRTRTC, SHRTYPE, SIPASGN, SIRASGQ, SIRCTAL,
 SIRTRAL, SLPHOUS, SLRBACS, SLRDADD, SLRFASM, SLRHLST, SLRROLL,
 SLRSCHD, SLRSCHE, SMPCPRG, SMRBCMP, SMRCMPL, SMRCRLT, SMRRLST,
 SOAIDEN, SOPAPPT, SOPLCCV, SOPLCPG, SOPSATS, SORAINF, SORCPLN,
 SOREMAL, SORHSRP, SORLCHG, SORPCSM, SORPGEO, SORSBSM, SORSGEO,
 SPRPDIR, SRREMAL, SRRENRH, SRRENRL, SRRINQR, SRRPREL, SRRSRIN,
 SRTLOAD, SRTPURG, SSPMFEE, SSPRDEF, SSPROLL, SSPSCHD, SSRATSQ,
 SSRRESV, SSRROLL, SSRSCMT, SSRSCPR, SSRSCRM, SSRSCUP, SSRSECT,
 SSRTALY, SSRUSEC, SURDELT, SURLOAD
TFRBILL, TFRDETL, TFRLATE, TFRRFND, TGPBILL, TGPHOLD, TGRAGES, TGRAPPL,
 TGRCDEL, TGRCLOS, TGRCOLC, TGRCSHR, TGRDELI, TGRDETC, TGRFEED,
 TGRMISC, TGRRCON, TGRRCPT, TGRUNAP, TRRAGES, TRRAPPL, TRRCOLL,
 TRRRCON, TRRUNAP, TRRUNPL, TSP1098, TSPISTA, TSPISTT, TSR1098,
 TSRBTOT, TSRCBIL, TSRDETL, TSRLATE, TSRLBOX, TSRRFND, TSRROLL,
 TSRSSUM, TSRTBIL, TSRTRAF, TSRTSUM, TVPREQA, TVRCRED
```

GORCCOL

Capture Table	Capture Columns
GOREMAL	GOREMAL_EMAIL_ADDRESS GOREMAL_PREFERRED_IND GOREMAL_STATUS_IND
GORIROL	GORIROL_ROLE GORIROL_ROLE_GROUP
SPBPERS	SPBPERS_BIRTH_DATE SPBPERS_LEGAL_NAME SPBPERS_NAME_PREFIX SPBPERS_NAME_SUFFIX SPBPERS_PREF_FIRST_NAME SPBPERS_SEX SPBPERS_SSN
SPRADDR	SPRADDR_ATYP_CODE SPRADDR_CITY SPRADDR_CNTY_CODE SPRADDR_NATN_CODE SPRADDR_STATUS_IND SPRADDR_STAT_CODE SPRADDR_STREET_LINE1 SPRADDR_STREET_LINE2 SPRADDR_STREET_LINE3 SPRADDR_ZIP
SPRIDEN	SPRIDEN_CHANGE_IND SPRIDEN_ENTITY_IND SPRIDEN_FIRST_NAME SPRIDEN_LAST_NAME SPRIDEN_MI
SPRTELE	SPRTELE_PHONE_AREA SPRTELE_PHONE_EXT SPRTELE_PHONE_NUMBER

GORCRUL

Capture Table	Capture Rule
SPRIDEN	SPRIDEN_CHANGE_IND is NULL SPRIDEN_ENTITY_IND IN (‘P’)

GORCMDD - Common Matching Data Dictionary Table

Table	Column	Element	Max. Lnth.	Override Lnth.	Allow Neg. Lnth.	On-line Ind.	Req. Element
SPRIDEN	SPRIDEN_ID	ID	9	Y	Y	Y	N
SPRIDEN	SPRIDEN_SEARCH_LAST_NAME	Last Name/ Non-Person Name	60	Y	N	Y	Y
SPRIDEN	SPRIDEN_SEARCH_FIRST_NAME	First Name	15	Y	N	Y	N
SPRIDEN	SPRIDEN_SEARCH_MI	Middle Name	15	Y	N	Y	N
SPRADDR	SPRADDR_STREET_LINE1	Street Line 1	30	Y	N	Y	N

GORCMDD - Common Matching Data Dictionary Table							
Table	**Column**	**Element**	**Max. Lnth.**	**Override Lnth.**	**Allow Neg. Lnth.**	**On-line Ind.**	**Req. Element**
SPRADDR	SPRADDR_CITY	City	20	Y	N	Y	N
SPRADDR	SPRADDR_STAT_CODE	State/ Province	3	N	N	Y	N
SPRADDR	SPRADDR_ZIP	Zip/ Postal Code	10	Y	N	Y	N
SPRADDR	SPRADDR_NATN_CODE	Nation	5	N	N	Y	N
SPRADDR	SPRADDR_CNTY_CODE	County	5	N	N	Y	N
SPRTELE	SPRTELE_PHONE_AREA	Telephone Area Code	3	Y	N	Y	N
SPRTELE	SPRTELE_PHONE_NUMBER	Telephone Number	7	Y	N	Y	N
SPBPERS	SPBPERS_SSN	SSN/ SIN/TIN	9	Y	Y	Y	N
SPBPERS	SPBPERS_BIRTH_DAY	Date of Birthday	2	N	N	Y	N
SPBPERS	SPBPERS_BIRTH_MON	Date of Birth Month	2	N	N	Y	N
SPBPERS	SPBPERS_BIRTH_YEAR	Date of Birth Year	4	N	N	Y	N
SPBPERS	SPBPERS_SEX	Gender	1	N	N	Y	N
GOREMAL	GOREMAL_EMAIL_ ADDRESS	Email	90	Y	N	Y	N

GORCTAB

Capture Table
GOREMAL
GORIROL
SPBPERS
SPRADDR
SPRIDEN

Capture Table
SPRTELE

GORDLUPAdd-In Data Lookup Repeating Table

Lookup Name	Lookup Description	Menu Seq. #	Position Control Ind.	HR Ind.	Finance Ind.	Load Function
F_FUND	Fund Code Lookup	1	N	N	Y	BANINST1.FBKD2SS.P_GET_FUND_LOOKUP
G_ORGN	Organization Code Lookup	2	Y	Y	Y	BANINST1.GOKDSSB.P_GET_ORGN_LOOKUP
F_PROG	Program Code Lookup	4	N	N	Y	BANINST1.FBKD2SS.P_GET_PROG_LOOKUP
F_ACCT	Account Code Lookup	3	N	N	Y	BANINST1.FBKD2SS.P_GET_ACCT_LOOKUP
F_LOCN	Location Code Lookup	6	N	N	Y	BANINST1.FBKD2SS.P_GET_LOCN_LOOKUP
F_ACTV	Activity Code Lookup	5	N	N	Y	BANINST1.FBKD2SS.P_GET_ACTV_LOOKUP
N_POSN	Position Lookup	7	N	Y	Y	BANINST1.NBKD2SB.P_LOOKUP_POSN
N_FISCYR	Fiscal Year Lookup	8	N	Y	Y	BANINST1.NBKD2SB.P_LOOKUP_FISCYR
N_ECLS	Employee Class Lookup	9	N	Y	Y	BANINST1.NBKD2SB.P_LOOKUP_ECLS
N_OBUD	Budget ID Lookup	10	N	N	Y	BANINST1.NBKD2SB.P_LOOKUP_OBUD
N_OBPH	Budget Phase Lookup	11	N	N	Y	BANINST1.NBKD2SB.P_LOOKUP_OBPH
N_EARN	Earnings Lookup	12	N	Y	N	BANINST1.NBKD2SB.P_LOOKUP_EARN
N_BDCA	Benefit/ Deduction Lookup	13	N	Y	N	BANINST1.NBKD2SB.P_LOOKUP_BDCA

GORDLUPAdd-In Data Lookup Repeating Table						
Lookup Name	**Lookup Description**	**Menu Seq. #**	**Position Control Ind.**	**HR Ind.**	**Finance Ind.**	**Load Function**
For all entries above, **Add-In Code** is `BUDGET`, **Financial Aid Indicator** is `N`, **Billcsh Indicator** is `N`, **Alumni Indicator** is `N`, **Student Indicator** is `N`.						

GORDMCL Display Mask Column Rules Table			
Block Name	**Column Name**	**Data Type**	**Data Length**
`GOVCMRT_MATCH`	`MATCH_BIRTH_DATE`	D	12
`GOVCMRT_MATCH`	`MATCH_CITY_STATE_ZIP`	C	30
`GOVCMRT_MATCH`	`MATCH_COUNTY_COUNTRY`	C	30
`GOVCMRT_MATCH`	`MATCH_EMAIL`	C	30
`GOVCMRT_MATCH`	`MATCH_ID`	C	9
`GOVCMRT_MATCH`	`MATCH_NAME`	C	99
`GOVCMRT_MATCH`	`MATCH_PHONE`	C	30
`GOVCMRT_MATCH`	`MATCH_SEX`	C	30
`GOVCMRT_MATCH`	`MATCH_SSN`	C	9
`GOVCMRT_MATCH`	`MATCH_STREET_LINE1`	C	30
`GOVCMRT_MATCH`	`MATCH_STREET_LINE2`	C	30
`GOVCMRT_MATCH`	`MATCH_STREET_LINE3`	C	30
`GOVCMRT_SUSPENSE`	`MATCH_BIRTH_DATE`	D	12
`GOVCMRT_SUSPENSE`	`MATCH_CITY`	C	30
`GOVCMRT_SUSPENSE`	`MATCH_EMAIL`	C	30
`GOVCMRT_SUSPENSE`	`MATCH_ID`	C	9
`GOVCMRT_SUSPENSE`	`MATCH_NAME`	C	99
`GOVCMRT_SUSPENSE`	`MATCH_NATN_CODE`	C	30
`GOVCMRT_SUSPENSE`	`MATCH_PHONE`	C	30
`GOVCMRT_SUSPENSE`	`MATCH_SEX`	C	30
`GOVCMRT_SUSPENSE`	`MATCH_SSN`	C	9
`GOVCMRT_SUSPENSE`	`MATCH_STATE`	C	3
`GOVCMRT_SUSPENSE`	`MATCH_STREET_LINE1`	C	30
`GOVCMRT_SUSPENSE`	`MATCH_ZIP`	C	9
For all entries above, **Display Object** is `GOAMTCH`, **Query Column** is null, and **Numeric Precision** is null.			

GORDPRP - Step Property Repeating Table		
Code	**Value**	**Description**
REQUIRED	TRUE	TRUE
REQUIRED	FALSE	FALSE
PICTURE	WIZARD_FUND	Wizard with Fund
PICTURE	WIZARD_BOOK	Wizard holding Book
PICTURE	WIZARD_ORGN	Wizard with Organization
MULTISELECT	TRUE	TRUE
MULTISELECT	FALSE	FALSE
FINDDISPLAYED	TRUE	TRUE
FINDDISPLAYED	FALSE	FALSE
PICTURE	WIZARD_QUESTION	Wizard with question marks
PICTURE	WIZARD_EXCLAM	Wizard with exclamation points
PICTURE	WIZARD_ACCT	Wizard with Account
PICTURE	WIZARD_PROG	Wizard with Program
PICTURE	WIZARD_LOCN	Wizard with Location
PICTURE	WIZARD_ACTV	Wizard with Activity
PICTURE	WIZARD_FLAG	Wizard holding a Finish Flag
PICTURE	WIZARD_CALENDAR	Wizard holding a calendar
PICTURE	WIZARD_CHART	Wizard behind a chart
PICTURE	WIZARD_BLOCK	Wizard with stack of blocks
PICTURE	WIZARD_AMOUNT	Wizard with money bags
PICTURE	WIZARD_EXCEL	Wizard holding spreadsheets

GORDSTE - Wizard Step Repeating Table			
Add-In Code	**Wizard Name**	**Step Name**	**Step Type**
BUDGET	DOWNLOAD	F_ACCT_FBBBLIN	ONEWIN
BUDGET	DOWNLOAD	F_ACCT_FGBOPAL	ONEWIN
BUDGET	DOWNLOAD	F_ACCT_FRRGRNL	ONEWIN
BUDGET	DOWNLOAD	F_ACTV_FBBBLIN	ONEWIN
BUDGET	DOWNLOAD	F_ACTV_FGBOPAL	ONEWIN
BUDGET	DOWNLOAD	F_ACTV_FRRGRNL	ONEWIN
BUDGET	DOWNLOAD	F_AMTTYPE_FBBBLIN	ONEWIN

GORDSTE - Wizard Step Repeating Table

Add-In Code	Wizard Name	Step Name	Step Type
BUDGET	DOWNLOAD	F_AMTTYPE_FGBOPAL	ONEWIN
BUDGET	DOWNLOAD	F_AMTTYPE_FRRGRNL	ONEWIN
BUDGET	DOWNLOAD	F_BUDGETDEV	OPTION
BUDGET	DOWNLOAD	F_BUDGID	ONEWIN
BUDGET	DOWNLOAD	F_BUDGPH	ONEWIN
BUDGET	DOWNLOAD	F_CMT_TYPE	OPTION
BUDGET	DOWNLOAD	F_COAS	ONEWIN
BUDGET	DOWNLOAD	F_FISCPERIOD	ONEWIN
BUDGET	DOWNLOAD	F_FISCYEAR	ONEWIN
BUDGET	DOWNLOAD	F_FUND_FBBBLIN	ONEWIN
BUDGET	DOWNLOAD	F_FUND_FGBOPAL	ONEWIN
BUDGET	DOWNLOAD	F_FUND_FRRGRNL	ONEWIN
BUDGET	DOWNLOAD	F_GRCODE	ONEWIN
BUDGET	DOWNLOAD	F_GRPERIOD	ONEWIN
BUDGET	DOWNLOAD	F_GRYEAR	ONEWIN
BUDGET	DOWNLOAD	F_LOCN_FBBBLIN	ONEWIN
BUDGET	DOWNLOAD	F_LOCN_FGBOPAL	ONEWIN
BUDGET	DOWNLOAD	F_LOCN_FRRGRNL	ONEWIN
BUDGET	DOWNLOAD	F_ORGN_FBBBLIN	ONEWIN
BUDGET	DOWNLOAD	F_ORGN_FGBOPAL	ONEWIN
BUDGET	DOWNLOAD	F_ORGN_FRRGRNL	ONEWIN
BUDGET	DOWNLOAD	F_PROG_FBBBLIN	ONEWIN
BUDGET	DOWNLOAD	F_PROG_FGBOPAL	ONEWIN
BUDGET	DOWNLOAD	F_PROG_FRRGRNL	ONEWIN
BUDGET	DOWNLOAD	F_REQ_COMPLETE	TEXT
BUDGET	DOWNLOAD	G_TABLE_NAME	OPTION
BUDGET	DOWNLOAD	N_BUDGID	ONEWIN
BUDGET	DOWNLOAD	N_BUDGPH	ONEWIN
BUDGET	DOWNLOAD	N_COAS	ONEWIN
BUDGET	DOWNLOAD	N_ECLS	ONEWIN
BUDGET	DOWNLOAD	N_FISCYR	ONEWIN

GORDSTE - Wizard Step Repeating Table			
Add-In Code	**Wizard Name**	**Step Name**	**Step Type**
BUDGET	DOWNLOAD	N_JOBS	OPTION
BUDGET	DOWNLOAD	N_JOBS_COAS	ONEWIN
BUDGET	DOWNLOAD	N_JOBS_COLS	TWOWIN
BUDGET	DOWNLOAD	N_JOBS_DATE	FREEFORMAT
BUDGET	DOWNLOAD	N_JOBS_INFO	OPTION
BUDGET	DOWNLOAD	N_ORGN	ONEWIN
BUDGET	DOWNLOAD	N_SOURCE	OPTION
BUDGET	UPLOAD	F_UPLOAD_BOOK	WKSHEET
BUDGET	UPLOAD	F_UPLOAD_BUDGID	ONEWIN
BUDGET	UPLOAD	F_UPLOAD_COAS	ONEWIN
BUDGET	UPLOAD	F_UPLOAD_FINISH	TEXT
BUDGET	UPLOAD	F_UPLOAD_HEADERS	ONEWIN
BUDGET	UPLOAD	F_UPLOAD_MAPPING	COLUMNMAP
BUDGET	UPLOAD	F_UPLOAD_MAPPING_ DUR	COLUMNMAP
BUDGET	UPLOAD	F_UPLOAD_PERM	OPTION
BUDGET	UPLOAD	F_UPLOAD_PHASE	ONEWIN
BUDGET	UPLOAD	F_UPLOAD_SEQNO	TEXT
BUDGET	UPLOAD	G_UPLOAD_TABLE	OPTION
BUDGET	UPLOAD	N_UPLD_BUDGID	ONEWIN
BUDGET	UPLOAD	N_UPLD_BUDGPH	ONEWIN
BUDGET	UPLOAD	N_UPLD_COAS	ONEWIN
BUDGET	UPLOAD	N_UPLD_FINISH	TEXT
BUDGET	UPLOAD	N_UPLD_FISCYR	ONEWIN
BUDGET	UPLOAD	N_UPLD_FTOT_BOOK	WKSHEET
BUDGET	UPLOAD	N_UPLD_FTOT_MAP	COLUMNMAP
BUDGET	UPLOAD	N_UPLD_HDERS_ WARNING	TEXT
BUDGET	UPLOAD	N_UPLD_PLBD_BOOK	WKSHEET
BUDGET	UPLOAD	N_UPLD_PLBD_MAP	COLUMNMAP
BUDGET	UPLOAD	N_UPLD_PLBD_MAP_ NOFIN	COLUMNMAP

GORDSTE - Wizard Step Repeating Table

Add-In Code	Wizard Name	Step Name	Step Type
BUDGET	UPLOAD	N_UPLD_PTOT_BOOK	WKSHEET
BUDGET	UPLOAD	N_UPLD_PTOT_MAP	COLUMNMAP
BUDGET	UPLOAD	N_UPLD_RTOT_BOOK	WKSHEET
BUDGET	UPLOAD	N_UPLD_RTOT_MAP	COLUMNMAP
BUDGET	VALIDATION	F_VAL_BOOK	WKSHEET
BUDGET	VALIDATION	F_VAL_BUDGID	ONEWIN
BUDGET	VALIDATION	F_VAL_COAS	ONEWIN
BUDGET	VALIDATION	F_VAL_FINISH	TEXT
BUDGET	VALIDATION	F_VAL_HEADERS	ONEWIN
BUDGET	VALIDATION	F_VAL_MAPPING	COLUMNMAP
BUDGET	VALIDATION	F_VAL_MAPPING_DUR	COLUMNMAP
BUDGET	VALIDATION	F_VAL_PERM	OPTION
BUDGET	VALIDATION	F_VAL_PHASE	ONEWIN
BUDGET	VALIDATION	F_VAL_SEQNO	TEXT
BUDGET	VALIDATION	G_VAL_TABLE	OPTION
BUDGET	VALIDATION	N_VAL_BUDGID	ONEWIN
BUDGET	VALIDATION	N_VAL_BUDGPH	ONEWIN
BUDGET	VALIDATION	N_VAL_COAS	ONEWIN
BUDGET	VALIDATION	N_VAL_FINISH	TEXT
BUDGET	VALIDATION	N_VAL_FISCYR	ONEWIN
BUDGET	VALIDATION	N_VAL_FTOT_BOOK	WKSHEET
BUDGET	VALIDATION	N_VAL_FTOT_MAP	COLUMNMAP
BUDGET	VALIDATION	N_VAL_HDERS_ WARNING	TEXT
BUDGET	VALIDATION	N_VAL_PLBD_BOOK	WKSHEET
BUDGET	VALIDATION	N_VAL_PLBD_MAP	COLUMNMAP
BUDGET	VALIDATION	N_VAL_PLBD_MAP_ NOFIN	COLUMNMAP
BUDGET	VALIDATION	N_VAL_PTOT_BOOK	WKSHEET
BUDGET	VALIDATION	N_VAL_PTOT_MAP	COLUMNMAP
BUDGET	VALIDATION	N_VAL_RTOT_BOOK	WKSHEET

GORDSTE - Wizard Step Repeating Table			
Add-In Code	**Wizard Name**	**Step Name**	**Step Type**
BUDGET	VALIDATION	N_VAL_RTOT_MAP	COLUMNMAP
BUDGET	UPLOAD	N_UPLD_SGRP	ONEWIN
BUDGET	VALIDATION	N_VAL_SGRP	ONEWIN

GORDSTP - Step Type Property Repeating Table			
Step Property Type	**Step Property Code**	**Locked**	**Required**
COLUMNMAP	REQUIREDCOLUMNS	Y	Y
COLUMNMAP	REQUIRED	Y	Y
ONEWIN	FINDDISPLAYED	Y	Y
TWOWIN	FINDDISPLAYED	Y	Y
WKSHEET	CAPTION	N	Y
WKSHEET	PICTURE	N	N
WKSHEET	SELECTIONPROC	Y	Y
WKSHEET	STORINGPROC	Y	Y
WKSHEET	REQUIRED	Y	Y
WKSHEET	MULTISELECT	Y	Y
TEXT	CAPTION_1	N	N
TEXT	CAPTION_2	N	N
TEXT	CAPTION_3	N	N
TEXT	PICTURE	N	N
FREEFORMAT	PICTURE	N	N
FREEFORMAT	CAPTION	N	Y
FREEFORMAT	SELECTIONPROC	Y	Y
FREEFORMAT	TEXTWIDTH	Y	Y
FREEFORMAT	TEXTHEIGHT	Y	Y
FREEFORMAT	STORINGPROC	Y	Y
TEXT	CAPTION_1_HT	N	N
FREEFORMAT	REQUIRED	Y	Y
FREEFORMAT	VALIDATIONPROC	Y	Y
TEXT	CAPTION_2_HT	N	N
TEXT	CAPTION_3_HT	N	N

GORDSTP - Step Type Property Repeating Table			
Step Property Type	**Step Property Code**	**Locked**	**Required**
TEXT	CAPTION_1_TOP	N	N
TEXT	CAPTION_2_TOP	N	N
TEXT	CAPTION_3_TOP	N	N
COLUMNMAP	POPULATIONPROC	Y	Y
ONEWIN	CAPTION	N	Y
ONEWIN	STORINGPROC	Y	Y
ONEWIN	REQUIRED	Y	Y
ONEWIN	POPULATIONPROC	Y	Y
ONEWIN	BOUNDCOLUMNS	Y	Y
ONEWIN	SELECTIONPROC	Y	Y
OPTION	OPTION_1	N	N
ONEWIN	PICTURE	N	N
OPTION	OPTION_2	N	N
OPTION	OPTION_3	N	N
OPTION	OPTION_4	N	N
OPTION	OPTION_5	N	N
OPTION	OPTION_6	N	N
OPTION	OPTION_7	N	N
OPTION	OPTION_1_KEY	N	N
OPTION	OPTION_2_KEY	N	N
OPTION	OPTION_3_KEY	N	N
OPTION	OPTION_4_KEY	N	N
OPTION	OPTION_5_KEY	N	N
OPTION	OPTION_6_KEY	N	N
OPTION	OPTION_7_KEY	N	N
OPTION	CAPTION	N	Y
OPTION	PICTURE	N	N
OPTION	STORINGPROC	Y	Y
OPTION	SELECTIONPROC	Y	Y
ONEWIN	COLUMNHEADERS	Y	Y
OPTION	OPTION_0	N	N

GORDSTP - Step Type Property Repeating Table			
Step Property Type	**Step Property Code**	**Locked**	**Required**
OPTION	OPTION_0_KEY	N	N
TWOWIN	BOUNDCOLUMNS	Y	Y
TWOWIN	CAPTION	N	Y
TWOWIN	PICTURE	N	N
TWOWIN	COLUMNHEADERS	Y	Y
TWOWIN	POPULATIONPROC	Y	Y
TWOWIN	STORINGPROC	Y	Y
TWOWIN	SELECTIONPROC	Y	Y
TWOWIN	REQUIRED	Y	Y
ONEWIN	MULTISELECT	Y	Y
OPTION	REQUIRED	Y	Y
COLUMNMAP	CAPTION	N	Y
COLUMNMAP	SELECTIONPROC	Y	Y
COLUMNMAP	COLUMNHEADERS	Y	Y
COLUMNMAP	STORINGPROC	Y	Y

GORDWIZ	Add-In Wizard Association Table					
Wizard Name	**Description**	**Add-In Code**	**Menu Seq.**	**Fin. Aid**	**Position Control**	**Billcsh**
DOWNLOAD	Download Wizard	BUDGET	1	N	Y	N
VALIDATION	Validation Wizard	BUDGET	2	N	Y	N
UPLOAD	Upload Wizard	BUDGET	3	N	Y	N

Wizard Name	HR	Finance	Advance	Student	Finish Function
DOWNLOAD	Y	Y	N	N	BANINST1.GOKDSSB.P_FINISH_DOWNLOAD
VALIDATION	Y	Y	N	N	BANINST1.GOKDSSB.P_FINISH_VALIDATION
UPLOAD	Y	Y	N	N	BANINST1.GOKDSSB.P_FINISH_UPLOAD

Wizard Name	Next Function	Unload Function
DOWNLOAD	BANINST1.GOKDSSB.P_NEXT_DOWNLOAD	BANINST1.GOKDSSB.P_UNLOAD_BUDGET
VALIDATION	BANINST1.GOKDSSB.P_NEXT_VALIDATION	BANINST1.GOKDSSB.P_UNLOAD_BUDGET
UPLOAD	BANINST1.GOKDSSB.P_NEXT_UPLOAD	BANINST1.GOKDSSB.P_UNLOAD_BUDGET

GOREQNM - Event Queue Name Definition Table			
Event Code	**Group Code**	**Target System Code**	**Status**
CHANGE_PERSON_NAME	CHGNAME	PIPELINE	I
CHANGE_PIN	PINCHANGE	PIPELINE	I
APPLICATION_RECEIVED	ID-MESSAGE	PIPELINE	I
GRADE_CHANGE	CHGGRADE	PIPELINE	I
GRADE_ROLL	GRADEROLL	PIPELINE	I
CHANGE_PERSON_ID	CHGPERSID	PIPELINE	I
CHANGE_MAJOR	CHGMAJOR	PIPELINE	I
SECTION_CANCELLED	ID-MESSAGE	PIPELINE	I
ADD_REGISTRATION	ADDREG	PIPELINE	I
ADD_SECTION	ADDSECTION	PIPELINE	I
PAFCHANGE	PAFCHANGE	WORKFLOW	I
NEWGIFT	NEWGIFT	WORKFLOW	I
WITHDRAWSTUDENT	WDSTUDENT	WORKFLOW	I
PSWDCHANGE	PSWDCHANGE	WORKFLOW	I
GRADECHG	GRADECHG	WORKFLOW	I
DOCAPPROVE	DOCAPPROVE	WORKFLOW	I
EDOCUMENT	EDOCUMENT	WORKFLOW	I
FAWITHDRAW	FAWITHDRAW	WORKFLOW	I
DROP_REGISTRATION	DROPREG	PIPELINE	I
ADD_NEW_STU_USER	ADDSTUDENT	PIPELINE	I
ADD_TEACH_ASSIGN	ADDTCHASG	PIPELINE	I
DELETE_TEACH_ASSIGN	DELTCHASG	PIPELINE	I
CHANGE_SECTION_NUM	CHGSECNUM	PIPELINE	I
CHANGE_COURSE_TITLE	CHGTITLE	PIPELINE	I

GOREQNM - Event Queue Name Definition Table			
Event Code	**Group Code**	**Target System Code**	**Status**
CHANGE_COURSE_DEPT	CHGDEPT	PIPELINE	I
DELETE_SECTION	DELSECTION	PIPELINE	I
ADD_NEW_FAC_USER	ADDFACULTY	PIPELINE	I
ADD_HOLD	ADDHOLD	PIPELINE	I
END_TERM	ENDTERM	PIPELINE	I
ADD_TERM	ADDTERM	PIPELINE	I
EMAIL_UPDATE	EMAILUPD	PIPELINE	A
EMAIL_INSERT	EMAILINS	PIPELINE	A
ICASSIGN	ICASSIGN	INTCOMP	I
ICENROLL	ICENROLL	INTCOMP	I
ICPERSON	ICPERSON	INTCOMP	I
ICSECTION	ICSECTION	INTCOMP	I
ICTERM	ICTERM	INTCOMP	I
CHANGE_MEETINGS	CHANGEMEET	PIPELINE	I
CHANGE_EMAIL_ID	CHGEMAILID	PIPELINE	I
CHANGE_SCHEDULE_CODE	CHGSCHCODE	PIPELINE	I
LDITERM	LDITERM	LDI	I
LDIPERSON	LDIPERSON	LDI	I
LDICOURSE	LDICOURSE	LDI	I
LDISECTION	LDISECTION	LDI	I
LDICOLLEGE	LDICOLLEGE	LDI	I
LDIDEPT	LDIDEPT	LDI	I
LDIXLGRP	LDIXLGRP	LDI	I
LDIXLMEM	LDIXLMEM	LDI	I
LDIENROLL	LDIENROLL	LDI	I
LDIASSIGN	LDIASSIGN	LDI	I

GORFDPI	FGAC PII Policy Table		
Table Name	**Column Name**	**Active**	**Driver SQL**
SPRIDEN	SPRIDEN_PIDM	N	gokfgac.f_find_pii_domain

GORRSQL - SQL Process Rules Table

Process Code	Rule Code	Seq. No.	Active	Start Date	Select From	Select Value
CARDHOLDER_ROLES	ALUMNUS	1	Y	19-OCT-05	FROM	SELECT
CARDHOLDER_ROLES	EMPLOYEE	1	Y	19-OCT-05	FROM	SELECT
CARDHOLDER_ROLES	STUDENT	1	Y	19-OCT-05	FROM	SELECT
HOUSING_ELIGIBILITY	STUDENT_ ENROLLED	1	Y	19-OCT-05	FROM	SELECT
INTCOMP	ALUMNI	1	Y	27-OCT-05	FROM	SELECT
INTCOMP	APPACCEPT	1	Y	27-OCT-05	FROM	SELECT
INTCOMP	APPLICANT	1	Y	27-OCT-05	FROM	SELECT
INTCOMP	DEVELOPMENT OFFICER	1	Y	27-OCT-05	FROM	SELECT
INTCOMP	EMPLOYEE	1	Y	27-OCT-05	FROM	SELECT
INTCOMP	FINANCE	1	Y	27-OCT-05	FROM	SELECT
INTCOMP	FRIENDS	1	Y	27-OCT-05	FROM	SELECT
INTCOMP	INTACCEPT	1	Y	27-OCT-05	FROM	SELECT
INTCOMP	PROSPECT	1	Y	27-OCT-05	FROM	SELECT
INTCOMP	STUDENT	1	Y	27-OCT-05	FROM	SELECT
INTCOMP	STUDENT	2	N	27-OCT-05	FROM	SELECT
INTCOMP	FACULTY	1	Y	27-OCT-05	FROM	SELECT
INTCOMP	FACULTY	2	N	27-OCT-05	FROM	SELECT

For all entries above, **Validated Indicator** is `Y`, and **End Date** is null. The **Where Clause** and **Parsed SQL** values for each row are shown in the tables below.

Rule Code	Seq. No.	Where Clause
ALUMNUS	1	`SELECT DISTINCT aprcatg_pidm FROM spriden, atvdonr, aprcatg WHERE spriden_entity_ind = 'P' AND spriden_change_ind IS NULL AND aprcatg_pidm = spriden_pidm AND atvdonr_code = aprcatg_donr_code AND atvdonr_alum_ind = 'Y'`
EMPLOYEE	1	`SELECT pebempl_pidm FROM pebempl WHERE NVL(pebempl_term_date, TRUNC(SYSDATE) + 1) > TRUNC(SYSDATE) AND NVL(pebempl_loa_beg_date, TRUNC(SYSDATE) - 1) < TRUNC(SYSDATE) AND pebempl_empl_status IN ('A', 'F', 'P')`

Rule Code	Seq. No.	Where Clause
STUDENT	1	SELECT sgbstdn_pidm FROM sgbstdn a, stvstst WHERE a.sgbstdn_stst_code = stvstst_code AND stvstst_reg_ind = 'Y' AND a.sgbstdn_term_code_eff = (SELECT MAX (b.sgbstdn_term_code_eff) FROM sgbstdn b WHERE b.sgbstdn_pidm = a.sgbstdn_pidm AND b.sgbstdn_term_code_eff <= :TERM)
STUDENT_ ENROLLED	1	SELECT sfbetrm_pidm FROM stvests, sfbetrm WHERE sfbetrm_term_code = :TERM AND stvests_code = sfbetrm_ests_code AND stvests_wd_ind = ‘N’
ALUMNI	1	SELECT DISTINCT aprcatg_pidm FROM aprcatg WHERE EXISTS (SELECT ‘X’ FROM atvdonr WHERE atvdonr_code = aprcatg_donr_code AND atvdonr_alum_ind = ‘Y’)
APPACCEPT	1	SELECT DISTINCT sarappd_pidm FROM sarappd WHERE sarappd_apdc_code IN (SELECT stvapdc_code FROM stvapdc WHERE stvapdc_stdn_acc_ind = ‘Y’)

Rule Code	Seq. No.	Where Clause
APPLICANT	1	SELECT DISTINCT p.saradap_pidm FROM saradap p WHERE NOT EXISTS (SELECT 'Y' FROM sarappd WHERE sarappd_pidm = p.saradap_pidm AND sarappd_term_code_entry = p.saradap_term_code_entry AND sarappd_appl_no = p.saradap_appl_no AND (sarappd_apdc_code IN (SELECT stvapdc_code FROM stvapdc WHERE stvapdc_inst_acc_ind = 'Y' AND stvapdc_stdn_acc_ind IS NULL) OR sarappd_apdc_code IN (SELECT stvapdc_code FROM stvapdc WHERE stvapdc_stdn_acc_ind IS NOT NULL))) AND NOT EXISTS (SELECT 'Y' FROM saradap s WHERE s.saradap_pidm = p.saradap_pidm AND s.saradap_term_code_entry = p.saradap_term_code_entry AND s.saradap_levl_code = p.saradap_levl_code AND EXISTS (SELECT 'Y' FROM sarappd WHERE sarappd_pidm = s.saradap_pidm AND sarappd_term_code_entry = s.saradap_term_code_entry AND sarappd_appl_no = s.saradap_appl_no AND (sarappd_apdc_code IN (SELECT stvapdc_code FROM stvapdc WHERE stvapdc_inst_acc_ind = 'Y' AND stvapdc_stdn_acc_ind IS NULL) OR sarappd_apdc_code IN (SELECT stvapdc_code FROM stvapdc WHERE stvapdc_stdn_acc_ind IS NOT NULL))))
DEVELOPMENT OFFICER	1	SELECT DISTINCT twgrrole_pidm FROM twgrrole WHERE twgrrole_role = 'DEVELOPMENTOFFICER'
EMPLOYEE	1	SELECT pebempl_pidm FROM pebempl, gtvsdax WHERE gtvsdax_external_code(+) = pebempl_ecls_code AND gtvsdax_internal_code_group(+) = 'INTCOMP' AND gtvsdax_internal_code(+) = 'LDIEMPEX' GROUP BY pebempl_pidm HAVING COUNT(gtvsdax_external_code) = 0
FINANCE	1	SELECT DISTINCT gobeacc_pidm FROM gobeacc, fobprof WHERE fobprof_user_id = gobeacc_username AND fobprof_web_access_ind = 'Y'
FRIENDS	1	SELECT DISTINCT aprcatg_pidm FROM aprcatg WHERE EXISTS (SELECT 'X' FROM atvdonr WHERE atvdonr_code = aprcatg_donr_code AND atvdonr_frnd_ind = 'Y')

Rule Code	Seq. No.	Where Clause
INTACCEPT	1	SELECT DISTINCT sarappd_pidm FROM sarappd, saradap WHERE saradap_pidm = sarappd_pidm AND saradap_term_code_entry = sarappd_term_code_entry AND saradap_appl_no = sarappd_appl_no AND sarappd_apdc_code IN (SELECT stvapdc_code FROM stvapdc WHERE stvapdc_inst_acc_ind = 'Y' AND stvapdc_stdn_acc_ind IS NULL) AND NOT EXISTS (SELECT 'Y' FROM saradap s WHERE s.saradap_pidm = sarappd_pidm AND s.saradap_levl_code = saradap_levl_code AND s.saradap_term_code_entry = sarappd_term_code_entry AND s.saradap_appl_no = sarappd_appl_no AND EXISTS (SELECT 'Y' FROM sarappd p WHERE p.sarappd_pidm = s.saradap_pidm AND p.sarappd_term_code_entry = s.saradap_term_code_entry AND p.sarappd_appl_no = s.saradap_appl_no AND p.sarappd_apdc_code IN (SELECT stvapdc_code FROM stvapdc WHERE stvapdc_stdn_acc_ind IS NOT NULL)))
PROSPECT	1	SELECT DISTINCT srbrecr_pidm FROM srbrecr WHERE NOT EXISTS (SELECT 'Y' FROM saradap WHERE saradap_pidm = srbrecr_pidm AND saradap_term_code_entry = srbrecr_term_code AND saradap_levl_code = srbrecr_levl_code)
STUDENT	1	SELECT DISTINCT a.sgbstdn_pidm FROM sgbstdn a, stvstst WHERE a.sgbstdn_stst_code = stvstst_code AND stvstst_reg_ind = 'Y' AND a.sgbstdn_term_code_eff IN (SELECT MAX(b.sgbstdn_term_code_eff) FROM sgbstdn b, sobterm c WHERE b.sgbstdn_pidm = a.sgbstdn_pidm AND b.sgbstdn_term_code_eff <= c.sobterm_term_code AND sobterm_profile_send_ind = 'Y' GROUP BY c.sobterm_term_code)
STUDENT	2	SELECT DISTINCT sfrstcr_pidm FROM sfrstcr WHERE sfrstcr_term_code IN (SELECT sobterm_term_code FROM sobterm WHERE sobterm_profile_send_ind = 'Y')

Rule Code	Seq. No.	Where Clause
FACULTY	1	SELECT DISTINCT a.sibinst_pidm FROM sibinst a, stvfcst WHERE a.sibinst_term_code_eff IN (SELECT MAX(b.sibinst_term_code_eff) FROM sibinst b, sobterm c WHERE b.sibinst_pidm = a.sibinst_pidm AND b.sibinst_term_code_eff <= c.sobterm_term_code AND sobterm_profile_send_ind = 'Y' GROUP BY c.sobterm_term_code) AND a.sibinst_fcst_code = stvfcst_code AND stvfcst_active_ind = 'A'
FACULTY	2	SELECT DISTINCT sirasgn_pidm FROM sirasgn WHERE sirasgn_term_code IN (SELECT sobterm_term_code FROM sobterm WHERE sobterm_profile_send_ind = ‘Y’)

Rule Code	Seq. No.	Parsed SQL
ALUMNUS	1	SELECT DISTINCT aprcatg_pidm FROM spriden, atvdonr, aprcatg WHERE spriden_entity_ind = 'P' AND spriden_change_ind IS NULL AND aprcatg_pidm = spriden_pidm AND atvdonr_code = aprcatg_donr_code AND atvdonr_alum_ind = 'Y'
EMPLOYEE	1	SELECT pebempl_pidm FROM pebempl WHERE NVL(pebempl_term_date, TRUNC(SYSDATE) + 1) > TRUNC(SYSDATE) AND NVL(pebempl_loa_beg_date, TRUNC(SYSDATE) - 1) < TRUNC(SYSDATE) AND pebempl_empl_status IN ('A', 'F', 'P')
STUDENT	1	SELECT sgbstdn_pidm FROM sgbstdn a, stvstst WHERE a.sgbstdn_stst_code = stvstst_code AND stvstst_reg_ind = 'Y' AND a.sgbstdn_term_code_eff = (SELECT MAX (b.sgbstdn_term_code_eff) FROM sgbstdn b WHERE b.sgbstdn_pidm = a.sgbstdn_pidm AND b.sgbstdn_term_code_eff <= :TERM)
STUDENT_ENROLLED	1	SELECT sfbetrm_pidm FROM stvests, sfbetrm WHERE sfbetrm_term_code = :TERM AND stvests_code = sfbetrm_ests_code AND stvests_wd_ind = ‘N’

Rule Code	Seq. No.	Parsed SQL
ALUMNI	1	SELECT DISTINCT aprcatg_pidm FROM aprcatg WHERE EXISTS (SELECT ‘X’ FROM atvdonr WHERE atvdonr_code = aprcatg_donr_code AND atvdonr_alum_ind = ‘Y’)
APPACCEPT	1	SELECT DISTINCT sarappd_pidm FROM sarappd WHERE sarappd_apdc_code IN (SELECT stvapdc_code FROM stvapdc WHERE stvapdc_stdn_acc_ind = ‘Y’)
APPLICANT	1	SELECT DISTINCT p.saradap_pidm FROM saradap p WHERE NOT EXISTS (SELECT ‘Y’ FROM sarappd WHERE sarappd_pidm = p.saradap_pidm AND sarappd_term_code_entry = p.saradap_term_code_entry AND sarappd_appl_no = p.saradap_appl_no AND (sarappd_apdc_code IN (SELECT stvapdc_code FROM stvapdc WHERE stvapdc_inst_acc_ind = ‘Y’ AND stvapdc_stdn_acc_ind IS NULL) OR sarappd_apdc_code IN (SELECT stvapdc_code FROM stvapdc WHERE stvapdc_stdn_acc_ind IS NOT NULL)))
		AND NOT EXISTS (SELECT ‘Y’ FROM saradap s WHERE s.saradap_pidm = p.saradap_pidm AND s.saradap_term_code_entry = p.saradap_term_code_entry AND s.saradap_levl_code = p.saradap_levl_code AND EXISTS (SELECT ‘Y’ FROM sarappd WHERE sarappd_pidm = s.saradap_pidm AND sarappd_term_code_entry = s.saradap_term_code_entry AND sarappd_appl_no = s.saradap_appl_no AND (sarappd_apdc_code IN (SELECT stvapdc_code FROM stvapdc WHERE stvapdc_inst_acc_ind = ‘Y’ AND stvapdc_stdn_acc_ind IS NULL) OR sarappd_apdc_code IN (SELECT stvapdc_code FROM stvapdc WHERE stvapdc_stdn_acc_ind IS NOT NULL))))

Rule Code	Seq. No.	Parsed SQL
DEVELOPMENT OFFICER	1	SELECT DISTINCT twgrrole_pidm FROM twgrrole WHERE twgrrole_role = 'DEVELOPMENTOFFICER'
EMPLOYEE	1	SELECT pebempl_pidm FROM pebempl WHERE gb_integ_config.f_exists('ELEARNING', 'LDIEMPEX', pebempl_ecls_code) = 'N' Background:
FINANCE	1	SELECT DISTINCT gobeacc_pidm FROM gobeacc, fobprof WHERE fobprof_user_id = gobeacc_username AND fobprof_web_access_ind = 'Y'
FRIENDS	1	SELECT DISTINCT aprcatg_pidm FROM aprcatg WHERE EXISTS (SELECT 'X' FROM atvdonr WHERE atvdonr_code = aprcatg_donr_code AND atvdonr_frnd_ind = 'Y')
INTACCEPT	1	SELECT DISTINCT sarappd_pidm FROM sarappd, saradap WHERE saradap_pidm = sarappd_pidm AND saradap_term_code_entry = sarappd_term_code_entry AND saradap_appl_no = sarappd_appl_no AND sarappd_apdc_code IN (SELECT stvapdc_code FROM stvapdc WHERE stvapdc_inst_acc_ind = 'Y' AND stvapdc_stdn_acc_ind IS NULL) AND NOT EXISTS (SELECT 'Y' FROM saradap s WHERE s.saradap_pidm = sarappd_pidm AND s.saradap_levl_code = saradap_levl_code AND s.saradap_term_code_entry = sarappd_term_code_entry AND s.saradap_appl_no = sarappd_appl_no AND EXISTS (SELECT 'Y' FROM sarappd p WHERE p.sarappd_pidm = s.saradap_pidm AND p.sarappd_term_code_entry = s.saradap_term_code_entry AND p.sarappd_appl_no = s.saradap_appl_no AND p.sarappd_apdc_code IN (SELECT stvapdc_code FROM stvapdc WHERE stvapdc_stdn_acc_ind IS NOT NULL)))

Rule Code	Seq. No.	Parsed SQL
PROSPECT	1	`SELECT DISTINCT srbrecr_pidm FROM srbrecr WHERE NOT EXISTS (SELECT 'Y' FROM saradap WHERE saradap_pidm = srbrecr_pidm AND saradap_term_code_entry = srbrecr_term_code AND saradap_levl_code = srbrecr_levl_code)`
STUDENT	1	`SELECT DISTINCT a.sgbstdn_pidm FROM sgbstdn a, stvstst WHERE a.sgbstdn_stst_code = stvstst_code AND stvstst_reg_ind = 'Y' AND a.sgbstdn_term_code_eff IN (SELECT MAX(b.sgbstdn_term_code_eff) FROM sgbstdn b, sobterm c WHERE b.sgbstdn_pidm = a.sgbstdn_pidm AND b.sgbstdn_term_code_eff <= c.sobterm_term_code AND sobterm_profile_send_ind = 'Y' GROUP BY c.sobterm_term_code)`
STUDENT	2	`SELECT DISTINCT sfrstcr_pidm FROM sfrstcr WHERE sfrstcr_term_code IN (SELECT sobterm_term_code FROM sobterm WHERE sobterm_profile_send_ind = ‘Y’)`
FACULTY	1	`SELECT DISTINCT a.sibinst_pidm FROM sibinst a, stvfcst WHERE a.sibinst_term_code_eff IN (SELECT MAX(b.sibinst_term_code_eff) FROM sibinst b, sobterm c WHERE b.sibinst_pidm = a.sibinst_pidm AND b.sibinst_term_code_eff <= c.sobterm_term_code AND sobterm_profile_send_ind = 'Y' GROUP BY c.sobterm_term_code) AND a.sibinst_fcst_code = stvfcst_code AND stvfcst_active_ind = 'A'`
FACULTY	2	`SELECT DISTINCT sirasgn_pidm FROM sirasgn WHERE sirasgn_term_code IN (SELECT sobterm_term_code FROM sobterm WHERE sobterm_profile_send_ind = ‘Y’)`

GORSQPA - SQL Process Parameter Table

	Process Code Code	Parameter Code
	CARDHOLDER_ROLES	TERM

GORSQPA - SQL Process Parameter Table		
	Process Code Code	**Parameter Code**
	HOUSING_ELIGIBILITY	TERM
	SEVIS	PIDM
	SEVIS	TERM

GORSSQL

Process	Rule	Parsed SQL Statement
IAM	IAM_GOBEACC_RULE	SELECT GOBEACC_USERNAME BANNERINB_USER FROM GOBEACC WHERE GOBEACC_PIDM =:PIDM

GTVCELG	Certification of Eligibility Table	
	I-20	I-20 Information
	I-94	I-94 Data
	IAP-66	International Information

GTVDADD	Add-In Code Validation Table	
	BUDGET	Spreadsheet Budgeting

GTVDIRO	Directory Options Validation Table	
	NAME	Name
	ADDR_PR	Permanent Address
	TELE_PR	Permanent Telephone
	ADDR_CP	Campus Address
	TELE_CP	Campus Telephone
	ADDR_OF	Office Address
	TELE_OF	Office Telephone
	TELE_FAX	Fax Number
	EMAIL	E-mail
	DEPT	Employee Department
	GRD_YEAR	Expected Graduation Year
	COLLEGE	College Affiliation
	TITLE	Employee Position Title

GTVDIRO	**Directory Options Validation Table**	
	ADDR_HO	Home Address
	CLASS_YR	Class Year
	ADDR_BU	Business Address
	MAIDEN	Maiden Name
	TELE_BU	Business Telephone
	PR_COLL	Preferred College
	TELE_HO	Home Telephone

GTVDPRP	**Step Property Validation Table**	
	REQUIREDCOLUMNS	Required Columns
	SELECTIONPROC	Selection Procedure
	OPTION_0	Option(0)
	OPTION_1	Option(1)
	OPTION_2	Option(2)
	OPTION_3	Option(3)
	OPTION_4	Option(4)
	OPTION_5	Option(5)
	OPTION_6	Option(6)
	OPTION_7	Option(7)
	OPTION_0_KEY	Option(0) - Key
	OPTION_1_KEY	Option(1) - Key
	OPTION_2_KEY	Option(2) - Key
	OPTION_3_KEY	Option(3) - Key
	OPTION_4_KEY	Option(4) - Key
	OPTION_5_KEY	Option(5) - Key
	OPTION_6_KEY	Option(6) - Key
	OPTION_7_KEY	Option(7) - Key
	PICTURE	Picture
	POPULATIONPROC	Population Procedure
	REQUIRED	Required
	TEXTWIDTH	Free Format Text Width
	MULTISELECT	Multiple Selections

GTVDPRP	**Step Property Validation Table**	
	STORINGPROC	Storing Procedure
	FINDDISPLAYED	Find Displayed
	TEXTHEIGHT	Free Format Text Height
	CAPTION_1_HT	Caption(1) Height
	VALIDATIONPROC	ValidationProc
	CAPTION_2_HT	Caption(2) Height
	CAPTION_3_HT	Caption(3) Height
	CAPTION_3_TOP	Caption(3) Top
	BOUNDCOLUMNS	Bounded Population Columns
	CAPTION_2_TOP	Caption(2) Top
	CAPTION	Caption
	CAPTION_1	Caption(1)
	CAPTION_2	Caption(2)
	CAPTION_3	Caption(3)
	COLUMNHEADERS	Column Headers
	CAPTION_1_TOP	Caption(1) Top

GTVDSTP	**Step Type Code Validation Table**	
	ONEWIN	One Window Step Type
	TEXT	Text Step Type
	OPTION	Option Step Type
	COLUMNMAP	Column Mapping Step Type
	TWOWIN	Two Window Step Type
	WKSHEET	Open Worksheets Step Type
	FREEFORMAT	Free Format Entry Step type

GTVDUNT	**Duration Unit Code Validation Table**		
Code	**Description**	**Number of Days**	**VR Message Number**
WEEK	Weeks	7	
MTHS	Months	31	

GTVEQNM	**Event Code Validation Table**	
	ADD_REGISTRATION	Add New Registration to CP

GTVEQNM	**Event Code Validation Table**	
	`ADD_NEW_STU_USER`	Add New Student User to CP
	`GRADE_CHANGE`	Grade Change
	`GRADE_ROLL`	Grade Roll
	`APPLICATION_RECEIVED`	Admissions Application Receipt
	`CHANGE_PIN`	Change PIN in CP
	`CHANGE_MAJOR`	Change Student Major in CP
	`CHANGE_PERSON_ID`	Change Person ID in CP
	`CHANGE_PERSON_NAME`	Change Person Name in CP
	`SECTION_CANCELLED`	Canceled Section Broadcast
	`PAFCHANGE`	Changes to the PAF on NOAEPAF
	`NEWGIFT`	A new Gift from a donor
	`WITHDRAWSTUDENT`	Student Withdrawal
	`PSWDCHANGE`	Password Change
	`GRADECHG`	A Students Grade Change
	`DOCAPPROVE`	Documents for Approval
	`EDOCUMENT`	Electronic Document
	`FAWITHDRAW`	Financial Aid Withdraw Student
	`DROP_REGISTRATION`	Drop Registration from CP
	`ADD_SECTION`	Add New Section to CP
	`ADD_TEACH_ASSIGN`	Add Teaching Assignment to CP
	`DELETE_TEACH_ASSIGN`	Delete Teaching Assignment
	`CHANGE_SECTION_NUM`	Change Section Number in CP
	`CHANGE_COURSE_TITLE`	Change Course Title in CP
	`CHANGE_COURSE_DEPT`	Change Course Department in CP
	`DELETE_SECTION`	Delete Section from CP
	`ADD_NEW_FAC_USER`	Add New Faculty User to CP
	`ADD_HOLD`	Add Hold Smart Event
	`END_TERM`	End Term in CP
	`ADD_TERM`	Add New Term to CP
	`EMAIL_UPDATE`	E-Mail Address Update

GTVEQNM	**Event Code Validation Table**	
	`EMAIL_INSERT`	E-Mail Address Insert
	`ICASSIGN`	IMS Faculty Assignment Event
	`ICENROLL`	IMS Enrolled Student Event
	`ICPERSON`	IMS Person Event
	`ICSECTION`	IMS Section Event
	`ICTERM`	IMS Term Event
	`CHANGE_MEETINGS`	Meeting Times in CP
	`CHANGE_EMAIL_ID`	EmailID change in CP
	`CHANGE_SCHEDULE_CODE`	Schedule code change in CP
	`LDITERM`	LDI Term Event
	`LDIPERSON`	LDI Person Event
	`LDICOURSE`	LDI Course Event
	`LDISECTION`	LDI Section Event
	`LDICOLLEGE`	LDI College Event
	`LDIDEPT`	LDI Department Event
	`LDIXLGRP`	LDI Cross Listed Group Event
	`LDIXLMEM`	LDI Cross Listed Member Event
	`LDIENROLL`	LDI Student Enrollment Event
	`LDIASSIGN`	LDI Faculty Assignment Event

GTVEQPC	**Group Code Validation Table**	
	`ID-MESSAGE`	ID and Message
	`PINCHANGE`	PIN Change
	`CHGMAJOR`	Change Student Major in CP
	`CHGNAME`	Change Person Name in CP
	`CHGPERSID`	Change Person ID in CP
	`ADDREG`	Add New Registration to CP
	`ADDSECTION`	Add New Section to CP
	`PAFCHANGE`	PAF Change on NOAEPAF
	`NEWGIFT`	A new Gift
	`WDSTUDENT`	Withdraw a Student
	`PSWDCHANGE`	Password Change

GTVEQPC	**Group Code Validation Table**	
	`GRADECHG`	Grade Change
	`DOCAPPROVE`	Documents for Approval
	`EDOCUMENT`	Electronic Document
	`FAWITHDRAW`	Financial Aid Withdraw Student
	`DROPREG`	Drop Registration from CP
	`ADDSTUDENT`	Add New Student User to CP
	`ADDTCHASG`	Add Teaching Assignment
	`DELTCHASG`	Delete Teaching Assignment
	`CHGSECNUM`	Change Section Number in CP
	`CHGTITLE`	Change Course Title in CP
	`CHGDEPT`	Change Course Department in CP
	`DELSECTION`	Delete Section from CP
	`ADDFACULTY`	Add New Faculty User to CP
	`CHGGRADE`	Grade Change
	`GRADEROLL`	Grade Roll
	`ENDTERM`	End Term in CP
	`ADDTERM`	Add New Term to CP
	`ADDHOLD`	Add New Hold in CP
	`EMAILUPD`	E-Mail Update
	`EMAILINS`	E-Mail Insert
	`ICASSIGN`	IMS Teaching Assignment Parms
	`ICENROLL`	IMS Student Enrollment Parms
	`ICPERSON`	IMS Person Parms
	`ICSECTION`	IMS Section Parms
	`ICTERM`	IMS Term Parms
	`CHANGEMEET`	Class Meetings Times in CP
	`CHGEMAILID`	Change EmailID in CP
	`CHGSCHCODE`	Change Schedule Code
	`LDITERM`	LDI Term Parms
	`LDIPERSON`	LDI Person Parms

GTVEQPC	**Group Code Validation Table**	
	LDICOURSE	LDI Course Parms
	LDISECTION	LDI Section Parms
	LDICOLLEGE	LDI College Parms
	LDIDEPT	LDI Department Parms
	LDIXLGRP	LDI Cross Listed Group Parms
	LDIXLMEM	LDI Cross Listed Member Parms
	LDIENROLL	LDI Student Enrollment Parms
	LDIASSIGN	LDI Faculty Assignment Parms

GTVEQPM - Parameter Code Validation Table	
MESSAGE	Message
ID	Person ID
EVENTTYPE	Event Type
$TEMPLATE	Template Name
SUBEVENTTYPE	Sub Event Type
CLEARTEXT/SCT.CREDENTIAL	Profile PIN Value
CLEARTEXT/CREDENTIAL	Campus Pipeline PIN Value
EmailID	E-Mail Address
UserName	Student/Faculty ID
Major	Student Major
LastName	Person Last Name
FirstName	Person First Name
SCT.ID	Student/Faculty ID
DONORNAME	Name of Donor
DONORPDC	Donor’s Primary Donor Category
GIFTAMT	the amount of the Gift
GIFTDATE	the date of the gift
GIFTNO	a Gift number
PIDM	pidm
TERM	Term Code
ENC_PASSWORD	Encrypted Oracle Password Code
ORACLE_USERNAME	Oracle Username Code

GTVEQPM - Parameter Code Validation Table	
DOCTYPE	Document Type
ACAT_CODE	PAF Approval Category Code
EFFECTIVE_DATE	effective date
EMPLOYEE_CLASS	Employee Class
EVENTNAME	Workflow Event Name (required)
PAF_ORIGINATOR_USERID	PAF Originator Oracle Userid
POSITION	Position
PRODUCTTYPE	Workflow Product Type (reqd)
TRANS_NO	PAF Transaction Number
TRANS_STATUS	PAF Transaction Status
WORKFLOWSPECIFICNAME	Workflow Specific Name (reqd)
DOCNUMBER	Document Number
AIDY	Aid Year Code
WITHDRAW_DATE	Withdraw Date
IDType	ID Type
SCT.Term.Description	Term Description
SCT.Course.Title	Course Title
SCT.Course.Term	Course Term
SCT.Course.Section	Course Section
SCT.Course.Instructor.ID	Course Instructor ID
SCT.Course.Instructor	Course Instructor Name
SCT.Course.Department	Course Department
Role	Profile Role
EnrolledCourse	Enrolled Course (CRN||Term)
ClearText/SCT.Credential	Profile PIN Value
ClearText/Credential	Profile PIN Value
DELIVERYTYPE	Message Delivery Type for CP
DisplayName	Display Name for CP
EnrollmentStatus.FullTime	Enrollment Status Description
MiddleName	Person Middle Name
SCT.Activity.Date	Activity Date
SCT.Course.Number	Course Number

GTVEQPM - Parameter Code Validation Table	
SCT.Hold.Description	Hold Type
SCT.Section.Title	Section Title
SCT.Subject.Code	Subject Code
url-0.TERM	Term Code for Smart Event
DATATYPE	Gen. Identifier for Event Type
G.DESCRIPTION.LONG	Long Group Name
G.DESCRIPTION.SHORT	Short Group Name
G.ENROLLCONTROL.ENROLLACCEPT	Accept Enrollment- Yes/No
G.ENROLLCONTROL.ENROLLALLOWED	Allow Enrolling- Yes/No
G.EXTENSION.DELIVERY	Course content delivery
G.GROUPTYPE.TYPEVALUE	Type Value
G.GROUPTYPE.TYPEVALUE.LEVEL	Type Value Level
G.ORG.ID	Org. Identifier
G.ORG.ORGNAM	Organization Name
G.ORG.ORGUNIT	Admin Unit, Math/English
G.RELATIONSHIP.LABEL	Nature of Group & SubGroup
G.RELATIONSHIP.MYRELATIONSHIP	1=Parent, 2=Child, 3=Other
G.RELATIONSHIP.SOURCEDID.ID	Group/SubGroup ID by System
G.SOURCEDID.ID	Group/SubGroup ID by System
G.TIMEFRAME.BEGIN	Available Participation Date
G.TIMEFRAME.END	Defines End Date
G.TRANSACTION	Rec type, 1 add/2 update/3 del
M.EXTENSION.MIDTERMRESULT.MODE	Desc of Midterm Grading Mode
M.MEMBER.IDTYPE	1=Person, 2=Group
M.MEMBER.ROLE.FINALRESULT.MODE	Desc of Final Result Mode
M.MEMBER.ROLE.ROLETYPE	01=Learner, 02=Instructor
M.MEMBER.ROLE.STATUS	1= Active, 2= Inactive
M.MEMBER.ROLE.SUBROLE	Further Defines Roles
M.MEMBER.ROLE.TRANSACTION	Rec type, 1 add/2 update/3 Del
M.MEMBER.ROLE.USERID	Person’s ID to Access Group
M.MEMBER.SOURCEDID.ID	ID of Org. or Source
M.SOURCEDID.ID	Person/Group/Sub Unique ID

GTVEQPM - Parameter Code Validation Table	
P.ADR.COUNTRY	Country
P.ADR.LOCALITY	Locality/City
P.ADR.PCODE	Postal Code
P.ADR.REGION	State or Province
P.ADR.STREET	Street Address
P.DEMOGRAPHICS.GENDER	Gender of Person
P.EMAIL	Email Address of Person
P.EXTENSION.USERROLE	User Role
P.EXTENSION.WEBCREDENTIAL	Web Credential
P.NAME.FN	Person’s name
P.NAME.N.FAMILY	Family name not last name
P.NAME.N.GIVEN	Given name
P.NAME.N.OTHER	Other name parts
P.NAME.N.PREFIX	Mr, Mrs, Ms, Dr etc
P.NAME.N.SUFFIX	Jr, III, Sr
P.NAME.NICKNAME	Preferred Name and format
P.SOURCEDID.ID	Person ID defined by Source
P.TEL	Phone Number of Person
P.TEL.TELTYPE	Phone# type, 1=Voice or 2=Fax
P.TRANSACTION	Rec type, 1 add/2 update/3 del
P.USERID	Person's access ID
SOURCE	Source of Event
M.EXTENSION.GRADABLE	Gradable Indicator
M.MEMBER.ROLE.COMMENTS	Member Comments
G.GROUPTYPE.SCHEME	Group type Coding Scheme
G.TIMEFRAME.BEGIN.RESTRICT	Allow Participation?- Yes/No
G.TIMEFRAME.END.RESTRICT	Defines Participation Ending
ClearText.Credential	Campus Pipeline Password Value
ClearText.SCT.Credential	Profile PIN Value
SourcedID.Source	Identifier for Source System
SourcedID.ID	Unique ID defined by Source
G.EXTENSION.DEL.RELATIONSHIP.LABEL	Nature of Group and SubGroup

GTVEQPM - Parameter Code Validation Table	
G.EXTENSION.DEL.RELATIONSHIP. MYRELATIONSHIP	1=Parent, 2=Child, 3=Other
G.EXTENSION.DEL.RELATIONSHIP. SOURCEDID.ID	Group/SubGroup ID by System
G.ATT.RECSTATUS	IMS Record Status
G.DESCRIPTION.FULL	Full Group Description Name
G.EXTENSION.LUMINISGROUP. DELIVERYSYSTEM	System delivering content
G.EXTENSION.LUMINISGROUP.EVENTS. RECURRINGEVENT.BEGINDATE	Event Begin Date
G.EXTENSION.LUMINISGROUP.EVENTS. RECURRINGEVENT.BEGINTIME	Event Begin Time
G.EXTENSION.LUMINISGROUP.EVENTS. RECURRINGEVENT.DAYSOFWEEK	Event Days of the Week
G.EXTENSION.LUMINISGROUP.EVENTS. RECURRINGEVENT.ENDDATE	Event End Date
G.EXTENSION.LUMINISGROUP.EVENTS. RECURRINGEVENT.ENDTIME	Event End Time
G.EXTENSION.LUMINISGROUP.EVENTS. RECURRINGEVENT.EVENTDESC	Event Description
G.EXTENSION.LUMINISGROUP.EVENTS. RECURRINGEVENT.LOCATION	TBA or Bldg w/ Room Number
G.EXTENSION.LUMINISGROUP.SORT	Term Sort Order
G.GROUPTYPE.TYPEVALUE.ATT.LEVEL	Group Type Level 1
G.RELATIONSHIP.ATT.RELATION	Group Relationship Attribute
G.TIMEFRAME.BEGIN.ATT.RESTRICT	Begin Restriction Attribute
G.TIMEFRAME.END.ATT.RESTRICT	End Restriction Attribute
M.MEMBER.ROLE.ATT.RECSTATUS	IMS Record Status
M.MEMBER.ROLE.ATT.ROLETYPE	Membership roletype
M.MEMBER.ROLE.EXTENSION. LUMINISROLE.GRADABLE	Gradable Indicator
M.MEMBER.ROLE.INTERIMRESULT. ATT.RESULTTYPE	Midterm result attribute
M.MEMBER.ROLE.INTERIMRESULT.MODE	Desc of Midterm Grading Mode
ONLINETOPIC	Y = publish to LMS
P.ATT.RECSTATUS	IMS Record Status

GTVEQPM - Parameter Code Validation Table	
P.EXTENSION.LUMINISPERSON.ACADEMICDEGREE	Faculty Academic Degree
P.EXTENSION.LUMINISPERSON.ACADEMICMAJOR	Student Academic Major
P.EXTENSION.LUMINISPERSON.ACADEMICTITLE	Faculty Academic Title
P.EXTENSION.LUMINISPERSON.CUSTOMROLE	Custom Person Role
P.INSTITUTIONROLE.ATT.INSTITUTIONROLETYPE	Person Institution Role
P.INSTITUTIONROLE.ATT.PRIMARYROLE	Person Primary Role
P.NAME.N.PARTNAME	Middle Name
P.NAME.N.PARTNAME.ATT.PARTNAMETYPE	Partname Type (Middlename)
P.TEL.ATT.TELTYPE	Telephone type attribute
P.USERID.ATT.PASSWORD	Userid password attribute
P.USERID.ATT.USERIDTYPE	Userid type attribute

GTVEQTS	Target System Code Validation Table	
	PIPELINE	Campus Pipeline
	WORKFLOW	SCT Workflow
	INTCOMP	SCT Integrator
	LDI	Luminis Data Integration

GTVLETR	Letter Process Letter Validation Table			
Code	**Duplicate**	**Description**	**Print Command**	**AlternateLetter Code**
MG_ACKN_LTR	Y	Matching Gift Acknowledgement		
EMP_MG_NOTICE	Y	Employee Notification of Match		
GIFT_RECEIPT	Y	Gift/Pledge Payment Receipt		
GIFT_ACKN_LTR	Y	Gift Acknowledgement Letter		

GTVLETR	**Letter Process Letter Validation Table**			
Code	**Duplicate**	**Description**	**Print Command**	**AlternateLetter Code**
`PLEDGE_ACKN`	Y	Pledge Acknowledgement Letter		
`DIRECTOR_THANKS`	N	Director’s Gift Thank you Ltr		
`PLEDGE_REMINDER`	Y	Special Pledge Reminder Letter		
`SPECIAL_GIFT`	N	Special Gift Acknowledgement		
`MAILING_LABEL`	N	Mailing Label	PL	
`DCSN`	N	Decision letters		
`MAJOR_GIFT`	N	Major Gift Acknowledgement		
`CORP_GIFT_ACKN`	Y	Corporate Gift Acknowledgement		
`FOUNDATION_ACKN`	Y	Foundation Gift Ackn Letter		
`FOUN_PLDG_ACKN`	Y	Foundation Pledge Ackn Letter		
`ANNUAL_FND_ACKN`	Y	Annual Fund Gift Ackn Letter		
`RECEIPT`	Y	Gift Receipt		
`FA_AWRD_W_COST`	Y	FA Award Letter with Costs		
`RESEARCH_PROFIL`	Y	Prospect Research Profile		
`WKBOOKLTR`	Y	Sample letter for G01C		
`ADM_APPL_ACKN`	N	Admissions Application Ackn		
`INQUIRY_THANKS`	Y	Thank you ltr all inq types		
`INF_REQ`	Y	Information Request Letter		
`DUES_ACKNOW`	Y	Dues Acknowledgement		`A/D_ACK_SPECIAL`

GTVLETR	**Letter Process Letter Validation Table**			
Code	**Duplicate**	**Description**	**Print Command**	**AlternateLetter Code**
`MEMBER_CARD`	Y	Membership Card		
`MEMB_DUES_ACK`	Y	Sample Membership Dues Letter		
`MEMBER_REMINDER`	Y	Membership Reminder Letter		
`MEMBER_RENEWAL`	Y	Membership Renewal Letter		
`MEMBER_RENEW_3`	Y	Membership 3rd Party Renewal		
`INVITATION`	Y	Invitation Letter		
`FA_TRACKING`	Y	Missing Inform. Letter -FINAID		
`AD_ACK_SPECIAL`	N	Acknowledgement of Special Gift		`AD_ACK_TWO`
`AD_ACK_TWO`	Y	Second Special Ackn of Gifts		
`AD_ACK_GIFTS`	Y	Gift Acknowledgement Letter		
`AD_QUIK_RECPT`	Y	Quick On line Gift Receipt		
`ADM_CHKL`	N	Admissions Checklist Letter		
`ADM_INT_1`	N	Admissions Interview 1 Letter		
`ADM_FA_INTEREST`	N	Financial Aid Interest Letter		
`T`	Y	t		
`TEST`	Y	t		
`HOUSING`	Y	Housing Information Letter		
`STEW_STUDENT`	Y	Stewardship Letter to Student		
`STEW_DESG_ID`	Y	Letter to Designation ID		

GTVLETR	Letter Process Letter Validation Table			
Code	Duplicate	Description	Print Command	AlternateLetter Code
COB_PCRNOTF_18M	Y	Cobra 18 Month Notification		
COB_PCRNOTF_36M	Y	Cobra 36 Month Notification		
COB_PCRLTRS_ENR	Y	Cobra Enrollment End Notices		
COB_PCRLTRS_LAT	Y	Cobra Late Notices		
COB_PCRLTRS_TER	Y	Cobra Termination Notices		
COB_PCRLTRS_PEX	Y	Cobra Pre-Expiration Notices		

GTVLFST	Learner Field of Study Type Validation Table	
	MAJOR	Major
	MINOR	Minor
	CONCENTRATION	Concentration

GTVMTYP	Meeting Type Validation page	
	CLAS	Classroom

GTVOBJT	VBS Object Code Validation Table	
	FORM	Oracle Forms module
	JOBS	Job Submission object
	MENU	Menu object
	MESSAGE	Menu Message object
	QUICKFLOW	QuickFlow object
	DLL	Dynamically Linked Library

GTVPARA - Letter Process Paragraph Validation Table		
Code	Description	Comment
STU_SAL	Student Salutation	Student Name, Addr, ID followed by ‘Dear xx,’
LABELDT	Define tables for Labels	Header paragraph containing define tables and invokes table 1.

GTVPARA - Letter Process Paragraph Validation Table

Code	Description	Comment
LABELS	Finaid label body	Body for labels
AWARD	Body of Finaid Award Letter	contains everything
APPADDR	Student's Name and Address	From the Student's Current Financial Aid Application
FA_NP	New page of letter	Start each letter at new page
AWARDS	Award Letter - Award Amounts	Award letter amount per term.
AWRDCLS	Award Letter Closing	Award letter closing with Sincerely, name and title of financial aid officer.
AWRDCST	Award Letter - Costs	Award letter costs, contributions, outside resources (with totals) and need
AWRDFAT	Award Letter - FAT Requirement	Financial Aid Transcript Requirements for Award Letter
AWRDHDR	Award Letter Heading	Award Letter Heading
AWRDINT	Award Letter Introduction	Award letter introduction with aid year desc
AWRDREQ	Award Letter - Requirements	Award letter requirements
AWRD_DT	Table Definitions for Award	Table Definitions for Financial Aid Award Letter
AWRD_NP	New Page for Award Letter	New Page with #RR for Award Letter
AWRD_TE	Table End for Award Letter	Table End for Financial Aid Award Letter
BASIC	basic constituent info	
GURADDR	Person Name/Address	Prints the person's name and address on the right margin.
GURINST	Institution Name/Address	Prints the institution address on the right margin of the page.
SRRCLOS	Recruiting Closing	Prints the titles of the person defined by the initial code.
SRRPRES	Presidential Greeting	Paragraph with presidential greeting message.
SRRINT1	Interview One Follow-up	Paragraph with Interview One Message.
SRRINT2	Interview Two Follow-up	Paragraph with Interview Two Message.

GTVPARA - Letter Process Paragraph Validation Table

Code	Description	Comment
SRRCNN1	College Night Follow-up	Paragraph with College Night Message.
GURCLOS	Closing	Prints “Sincerely” and spacing on the bottom of the page.
GURLABL	Mailing Label Name/Address	Paragraph with name and address to be used as mailing label.
CALCVAR	Calculated Variables	Calculated Variables in Financial Aid Award Letter
FADIR	Financial Aid Director	Financial Aid Director’s Name
FA_HEDR	Financial Aid Letter Header	Header for Financial Aid Tracking Letter
FA_SALU	Financial Aid Salutation	Financial Aid Salutation Paragraph
FUNDMSG	Message Text for Funds	Message text associated with selected fund codes.
GURSALU	Salutation	Prints the date on right margin and “Dear xx” on the left margin.
INAME	Name, address of Institution	Institution Name and Address printed in the center of the letter, 1 line per address field except city/state/zip
TABLE1	Invoke Table 1	Invoke Table 1 in Financial Aid Award Letter
TRACK12	Tracking Paragraph w. Msgs	Tracking Paragraph using messages from RORMESG table
LABEL	File Labels	Internal File labels
MLABEL	Mailing Label - Name / Address	Paragraph with name and address to be used as a mailing label
AK_RCPT	A/D Gift Ack. Receipt	Alumni/Development gift acknowledgement receipt.
TRACK	Financial Aid Req. Tracking	Body of Financial Aid Requirements Tracking Letter
TRCK_DT	Table Definitions for Tracking	Table Definitions for Financial Aid Tracking Letter

GTVPARA - Letter Process Paragraph Validation Table

Code	Description	Comment
RESEARC	info from prospect research	
ACK_TAB	Ack tables 1-3	Gift Acknowledgement letter table definition.
AKGBODY	Alumni/Dev ack gift body	Gift acknowledgement thank you with amount, campaigns.
ANAMEAD	Alumni Ack Const. addr name	Acknowledgment address name for constituent.
ANAMESL	A/D Ack. first name salutation	Alumni Development name salutation for acknowledgements.
AORGNNM	Alumni Ack org addr name	Acknowledgement address name for organization.
AORGNSL	A/D Ack. orgn. name salutation	Alumni Development org primary name salutation for acknowledgements.
APREFAD	Alumni Ack preferred address	Preferred address type from constituent page.
AKGCLAS	Alumni/Dev ack Class paragraph	Gift acknowledgement preferred class reference.
A_ETAB	End table	
A_ITAB1	Invoke table 1	Alumni Invoke table 1.
SIGN	Signature block	Prints Sincerely, name, and title for initials used with letter
AKGSIGN	Alumni/Dev ack signature	Gift acknowledgement signature
AK_RAMT	A/D Gift Ack. Receipt amount	Alumni/Development gift acknowledgement receipt amt, date, gift number.
LTRDATE	Letter Date	
TOPPAGE	Top of Page	
ACPT_TE	Ends tables for Acceptance	End table commands for acceptance letters
CHKLLST	List of Checklist items	Lists each checklist items and it received date after body of letter
INFADDR	Informal name, address, & salut	Prints the name, street, city, state, zip and salutation without a title (i.e. Mr., Dr.)

GTVPARA - Letter Process Paragraph Validation Table

Code	Description	Comment
INQUIRY	Body of the Inquiry thanks ltr	Prints the body & closing of the Inquiry_thanks letter with use of most recruiting variables
ADMACKL	Admissions Application Ackl	Admissions Application Acknowledgement, including missing Checklist Items, if any
FRMADDR	Formal Name, Address & Sal	Prints the formal name with prefix and suffix, full address and salutation
INADDRS	Institution Name & address	Prints and centers the institution name, address, & phone #
ACCEPT	Admissions Acceptance Para	Body of the Admissions Acceptance letter
TB_RECR	Table for Recruiting Letter	Table to Indent Institution Name and Signature Variable
CLOSING	Admissions/Recruiting Closing	Prints Sincerely, name, and title for initials
INFOREQ	Information Request	Body of information request letter
GURPERS	Person Name/Address	Prints the persons name and address on the left margin of the page.
WKBOOK1	Workbook Para 1 (Inside Addr)	This is the first paragraph used in the Letter Generation Textbook. Notice that this long comment scrolls. The paragraph includes today's date, name and address. It includes examples of the use of the ^IFNULL command.
ACPT_DT	Table definitions for Accept	All table definitions used for Acceptance
OPENHOU	Body of the Open House Letter	Prints the body of the Open House Invitation Letter
HEADER	Use as 1st Paragraph	Forces new page. Prevents page creep.
CLOSE	Sharon Weinberg Signature	Includes skip after body, closing and signature

GTVPARA - Letter Process Paragraph Validation Table

Code	Description	Comment
APPOINT	Recruiting Appointment Letter	Includes appointment type, date and times.
DUE_ACK	Dues Acknowledgment Body	
DUE_TAB	Dues Acknowledgment Tables	
MEMB_CD	Membership Card Paragraph	
MEMB_TB	Membership Define Tables	
MEM_3TB	Renewal Letter -3rd Party Tabl	Tables for 3rd Party Renewal Letter
CHKLBDY	Admissions Checklist	Body of Admissions Checklist letter
MEM_REM	Reminder Letter Paragraph	
MEM_REN	Renewal Letter Paragraph	
MEM_RN3	Renewal Letter - 3rd Party	
ACK_TDF	Table Definitions for Gift Ack	Gift Acknowledgement letter table definition.
ACK_NPG	New Page Command	
ACK_LIN	Line Count for Page	
ACK_DTE	Letter Date	
ACK_NAD	Name and Address for Ack.	Person or Org Name and Address
ACK_SAL	Person/Org Salutations	Person or organization salutations for acknowledgement/receipt
ACK_BDY	Body of Acknowledgement Letter	
INVITE	Invitation for a Function	
HEADDTE	Letter Date	Prints current date on left side of page
NEWPAGE	New page for letter	Start each letter at new page
WKBOOK2	Workbook Para 2 (Inf Sal)	Workbook paragraph 2, which contains an informal salutation followed by a comma.
WKBOOK3	Workbook Para 3 (Person Verf)	Workbook paragraph 3, which contains the body of the letter (current Id, gender and marital status).

GTVPARA - Letter Process Paragraph Validation Table		
Code	**Description**	**Comment**
ENCL	Enclosures Paragraph	
T	t	
MNYROOM	More than 1 Roommate Info	Paragraph to Print Address/ Phone Info for More than 1 Roommate
MANYALT	Many Roommates Alternate Para	Alternate Paragraph Formatting for Printing Many Roommates Info
ONERALT	One Roommate Alternate Para	Alternate Paragraph Formatting for Printing One Roommate Info
CLSHOUS	Closing for Housing Letter	Closing for Housing Letter
SNGLERM	Single Room Housing Info	Paragraph to Print Single Room Housing Information
ONEROOM	One Roommate Housing Info	Paragraph to Print Address/ Phone Info for One Roommate Only
INTHOUS	Introduction to Housing Letter	Introduction to Housing Letter
TB_HOUS	Table for Housing Letter	Table Definitions for Housing Letter
STEW2	Stewardship to student	Stewardship letter to student
STEW1	Stewardship to Desg ID	Stewardship letter to designation ID
COB_TAB	Cobra Tables	
COB_NPG	Cobra New Page	
COB_HDR	Cobra Header	
COB_NTA	Cobra Notification Letr Para 1	
COB_NTB	Cobra Notification Letr Para 2	
COB_NTC	Cobra Notification Letr Para 3	
COB_NTD	Cobra Notification Letr Para 4	
COB_NTE	Cobra Notification Letr Para 5	
COB_NTF	Cobra Notification Letr Para 6	
COB_SSD	Cobra SS Procedure	
COB_SSP	Cobra SS Procedure	
COB_ELE	Cobra Election page	

GTVPARA - Letter Process Paragraph Validation Table		
Code	**Description**	**Comment**
`COB_ENR`	Cobra Enrollment End Notices	
`COB_LAT`	Cobra Late Notices	
`COB_TER`	Cobra Termination Notices	
`COB_PEX`	Cobra Pre-Expiration Notices	
`COB_HLP`	Cobra Admin Contact Address	

GTVPARS	**Partition Code Validation Table**		
Code	**Description**	**Scheduler Number**	**Campus Code**
0	Default Partition	0	

GTVPROC	**Process Name Validation Table**	
	WEBCCEPRTREQ	Web Credit Card Enrollment Verification Charge Process
	WEBCCREGFEES	Web Credit Card Registration Fees Process
	WEBCCAPPFEES	Web Credit Card Application Fees Process
	WEBCCGRADAPP	Web Credit Card Graduation Application Process

GTVRRAC	**Regulatory Race Validation Table**	
	1	American Indian or Alaskan Native
	2	Asian
	3	Black or African American
	4	Native Hawaiian and Other Pacific Islander
	5	White
	For all entries above, **Data Origin** is null.	

GTVSCHS	**Scheduling Status Code Validation Form**	
	NSM	Class needs a room assignment.

GTVSCHS	Scheduling Status Code Validation Form	
	1SM	Class needs a room assignment and has a preferred first choice room indicated in the **Room Name** field. This code limits the initial pool of candidate rooms in the assignment algorithm.
	WSM	Class needs a room assignment and must be assigned with the preceding NSM or 1SM record to the same room at the same time (cross-listed).
	RSM	Class is related to the preceding NSM or 1SM record and must be assigned to the same room but not at the same days/time.
	NXM	Class needs a room assignment and can share a room with another class whose times overlap with it (can be doubled-booked).
	1XM	Class needs a room assignment, has a preferred first choice room indicated in the **Room Name** field, and can share a room with another class whose times overlap with it (can be double-booked).
	RXM	Class is related to the previous NXM or 1XM record and must be assigned to the same room at the same or overlapping times.
	ASM	Class has a room assignment that was made manually or in another system, such as the student information system.

GTVSCHS	Scheduling Status Code Validation Form	
	AXM	Class has a room assignment that was made manually or in another system, and the class time span overlaps part of all of the time span of another class assigned to the same room (double-booking or intentional conflict).
	HSM	This is a set of home cross-listed classes pre-assigned to the same room at identical days and times.
	VSM	This is a set of visitor cross-listed classes pre-assigned to the same room at identical days and times.
	5SM	Schedule25 assigned the class a room during a previous run.
	5XM	Schedule25 assigned the class a room, and it is double-booked with another class.

GTVSQPR	SQL Process Code Validation Table		
Code	**Description**	**Start Date**	**End Date**
`CARDHOLDER_ROLES`	Cardholder roles	19-Oct-05	
`HOUSING_ELIGIBILITY`	Housing Integration, Eligibility Roles	19-Oct-05	
`INTCOMP`	Integration roles	27-Oct-05	
`SEVIS`	SEVIS Processing	2-Oct-03	

GTVSQRU	SQL Rule Code Validation Table		
Code	**Description**	**Start Date**	**End Date**
`ALUMNUS`	Alumnus Role	19-Oct-05	
`EMPLOYEE`	Employee Role	19-Oct-05	
`STUDENT`	Student Role	19-Oct-05	
`STUDENT_ENROLLED`	Student with enrollment in given term	19-Oct-05	
`ALUMNI`	Alumni Role	27-Oct-05	
`FACULTY`	Faculty Role	27-Oct-05	

GTVSQRU	SQL Rule Code Validation Table		
Code	**Description**	**Start Date**	**End Date**
FRIENDS	Friend Role	27-Oct-05	
FINANCE	Finance Role	27-Oct-05	
DEVELOPMENTOFFICER	Development Officer Role	27-Oct-05	
PROSPECT	Prospect Role	27-Oct-05	
APPLICANT	Applicant Role	27-Oct-05	
INTACCEPT	Institution Accept Role	27-Oct-05	
APPACCEPT	Applicant Accept Role	27-Oct-05	
IAM_GOBEACC_RULE	GOBEACC attributes for IAM		

GTVSVAP	SEVIS Auto-populate Code Validation Table	
	GOBSEVS_VTYP_CODE	Visa type code
	GOBSEVS_BIRTH_NATN_CODE	Birth nation code
	GOBSEVS_LEGAL_NATN_CODE	Legal nation code
	GOBSEVS_PROGRAM_BEGIN_DATE	Program begin date
	GOBSEVS_PROGRAM_END_DATE	Program end date
	GOBSEVS_PROGRAM_ENROLL_DATE	Program enroll date
	GOBSEVS_ACADEMIC_TERM_MONTHS	Academic term in months
	GOBSEVS_TUITION_EXPENSE	Tuition expense
	GOBSEVS_PERSONAL_FUNDS	Personal funds
	GOBSEVS_SESSION_START_DATE	Session start date
	GOBSEVS_SESSION_END_DATE	Session end date
	GOBSEVS_SVEL_CODE	Education level code
	GOBSEVS_SVEL_COMMENT	Education level comment
	GOBSEVS_MAJR_CODE	Major code
	GOBSEVS_STUDY_LENGTH	Length of study in months
	GOBSEVS_LIVING_EXPENSES	Living expenses
	GOBSEVS_SVFT_CODE	Drop below full-time status code
	GOBSEVS_AUTH_START_DATE	Authorized start date
	GOBSEVS_COMPLETION_REMARKS	Completion remarks
	GOBSEVS_NEW_PROGRAM_END_DATE	New program end date

GTVSVAP	**SEVIS Auto-populate Code Validation Table**	
	`GOBSEVS_DA_PROGRAM_END_DATE`	Deferred attendance program end date
	`GOBSEVS_DA_PROGRAM_START_DATE`	Deferred attendance program start date
	`GOBSEVS_DISC_ACTION_TEXT`	Disciplinary action comment
	`GOBSEVS_EXTEND_END_DATE`	Extension end date
	`GOBSEVS_SVCR_CODE`	Creation reason code
	`GOBSEVS_SVCR_COMMENT`	Creation reason comment
	`GOBSEVS_SVTR_CODE`	Termination code
	`GOBSEVS_TERMINATE_DATE`	Termination date
	`GOBSEVS_OTHER_INFRACT_COMMENT`	Infraction comment
	`GOBSEVS_SVEP_CODE`	End program code
	`GOBSEVS_END_PROGRAM_EFF_DATE`	End program effective date
	`GOBSEVS_EV_FORM_NUMBER`	Exchange Visitor page number
	`GOBSEVS_SVRP_CODE`	Reprint reason code
	`GOBSEVS_REPRINT_REASON_ COMMENT`	Reprint reason comment
	`GOBSEVS_PRINT_REQUEST_IND`	Print request indicator
	`GOBSEVS_DEPENDENT_EXPENSES`	Dependent expenses
	`GOBSEVS_OTHER_FUNDS`	Other funds
	`GOBSEVS_OTHER_FUNDS_COMMENT`	Other funds comment
	`GOBSEVS_OTHER_EXPENSES`	Other expenses
	`GOBSEVS_OTHER_EXP_COMMENT`	Other expenses comment
	`GOBSEVS_AUTH_END_DATE`	Authorization end date
	`GOBSEVS_NEW_PROGRAM_START_ DATE`	Program initial start date for continuing EV
	`GOBSEVS_SVPC_CODE`	Program code
	`GOBSEVS_SVSC_CODE`	Subject code
	`GOBSEVS_SVSC_COMMENT`	Subject code comment
	`GOBSEVS_COMMUTER_IND`	Commuter indicator
	`GOBSEVS_ENG_PROF_REQ_IND`	English proficiency required indicator
	`GOBSEVS_ENG_PROF_MET_IND`	English proficiency met indicator
	`GOBSEVS_ENG_PROF_REASON`	English proficiency comment

GTVSVAP	SEVIS Auto-populate Code Validation Table	
	GOBSEVS_CRIMINAL_CONVICT_IND	Criminal conviction indicator
	GOBSEVS_ADMISSION_NUMBER	Admission number
	GOBSEVS_DRIVERS_LIC_NUMBER	DriverNULLs license number
	GOBSEVS_STAT_CODE_DRIVERS_LIC	DriverNULLs license state of issue
	GOBSEVS_TIN	Taxpayer identification number
	GOBSEVS_MAJR_CODE_2	Secondary major code
	GOBSEVS_MAJR_CODE_MINR	Minor code
	GOBSEVS_SCHOOL_FUNDS	School funds
	GOBSEVS_SCHOOL_FUNDS_COMMENT	School funds comment
	GOBSEVS_EMPLOYMENT_FUNDS	Employment funds
	GOBSEVS_FUNDING_COMMENT	Funding comment
	GOBSEVS_SVFT_COMMENT	Drop below fill time status comment
	GOBSEVS_SVEP_COMMENT	End program comment
	GOBSEVS_DA_COMMENT	Deferred attendance comment
	GOBSEVS_PASSPORT_NUMBER	Passport number
	GOBSEVS_NATN_CODE_PASSPORT	Country issuing passport
	GOBSEVS_PASSPORT_EXPIRE_DATE	Passport expiry
	GOBSEVS_VISA_NUMBER	Visa number
	GOBSEVS_SVCP_CODE	Consular post code
	GOBSEVS_VISA_EXPIRE_DATE	Visa expiry
	GOBSEVS_PENT_CODE	Port of entry code
	GOBSEVS_PENT_COMMENT	Port of entry comment
	GOBSEVS_ENTRY_DATE	Entry date
	GOBSEVS_RFC_COMMENT	Resume full course comment
	GOBSEVS_EDIT_PROGRAM_COMMENT	Edit program comment
	GOBSEVS_PROGRAM_SPONSOR_FUNDS	Program sponsor funds
	GOBSEVS_GOVT_ORG_FUNDS	Government organization 1 funds
	GOBSEVS_SVGO_CODE	Government organization 1 code
	GOBSEVS_GOVT_ORG_FUNDS_2	Government organization 2 funds

GTVSVAP	SEVIS Auto-populate Code Validation Table	
	GOBSEVS_SVGO_CODE_2	Government organization 2 code
	GOBSEVS_INTL_ORG_FUNDS	International organization 1 funds
	GOBSEVS_SVIO_CODE	International organization 1 code
	GOBSEVS_INTL_ORG_FUNDS_2	International organization 2 funds
	GOBSEVS_SVIO_CODE_2	International organization 2 code
	GOBSEVS_EV_GOVT_FUNDS	Funds from the EV's government
	GOBSEVS_BINATION_FUNDS	Binational funds
	GOBSEVS_OTHER_ORG_FUNDS	Other organization funds
	GOBSEVS_PROGRAM_START_IND	Program start indicator
	GOBSEVS_EXTEND_PROGRAM_COMMENT	Comment on program extension
	GOBSEVS_AMEND_PROGRAM_COMMENT	Comment on program amendment
	GOBSEVS_MATRICULATION_CDE	Matriculation code
	GOBSEVS_BIRTH_CITY	Birth city
	GOBSEVS_EDIT_BIO_COMMENT	Edit biographical data comment
	GOBSEVS_NATN_CODE_PERM_RES	Country of permanent residency
	GOBSEVS_FIN_SUPPORT_COMMENT	Financial support comment
	GOBSEVS_SVIT_CODE	Infraction type code
	GOBSEVS_SVCC_CODE	Category code

For all entries above, **Start Date** is 2-Oct-03 and **End Date** is null.

GTVSVBA	SEVIS Business Action Code Validation Table			
Code	**Description**	**Procedure Name**	**Start Date**	**End Date**
CREATE_STUDENT	Create student for SEVIS processing	goksvsq.p_create_student	2-Oct-03	
CREATE_EV	Create Exchange Visitor for SEVIS processing	goksvsq.p_create_ev	2-Oct-03	

GTVSVCC	SEVIS Exchange Visitor Program Category Code Validation Table	
	1A	Student Secondary
	1B	Student Associate

GTVSVCC	SEVIS Exchange Visitor Program Category Code Validation Table	
	1C	Student Bachelors
	1D	Student Masters
	1E	Student Doctorate
	1F	Student Non-degree
	2A	Trainee (specialty)
	2B	Trainee (non-specialty)
	3	Teacher
	4	Professor
	5	International Visitor
	6	Alien Physician
	7	Government Visitor
	8	Research Scholar
	9	Short-term scholar
	10	Specialist
	11	Camp Counselor
	12	Summer work/travel
	13	Aupair

GTVSVCR - SEVIS Creation Reason Code Validation Table

Code	Description	Usage Indicator
S	Change of Status	1
I	Initial	1
C	INAC 5/05 Continued Attendance	1
T	INAC 5/05 Transfer	1
D	INAC 5/05 Dependent	1
R	INAC 5/05 Reinstatement	1
O	INAC 5/05 Other	1
1	INACT 1/03--Begin New Program	2
2	INACT 1/03--Continuing EV	2
3	INACT 1/03Transffrom non-SEVIS	2

GTVSVCR - SEVIS Creation Reason Code Validation Table		
Code	**Description**	**Usage Indicator**
4	INACT 1/03--Reinstatement	2
CONT	INAC 5/05 Continuing	2
NEW	New	2

GTVSVDT	**SEVIS Dependent Termination Code Validation Table**	
	1	Conviction of a Crime
	2	Death
	3	Child Over 21
	4	Divorce
	5	Unauthorized Employment
	6	Principal Status Terminated
	7	INAC 5/5 271 Days Post ProgEnd
	8	INAC 5/5 271 Days Post PrinEnd
	9	Other
	10	Principal Status Completed
	11	INAC 5/5 Terminated J-1 Visa
	12	INAC 5/5 Completed J-1 Visa

GTVSVEL	**SEVIS Educational Level Code Validation Table**	
	1	Primary
	2	Secondary
	3	Associate
	4	Bachelors
	5	Masters
	6	Doctorate
	7	Language Training
	8	High School
	9	Flight School
	10	Other Vocational School
	11	Other

GTVSVEP	**SEVIS End EV Program Reason Code Validation Table**	
	COMP	Completed
	1	INACT 1/03Withdrawal From Prog
	2	INACT 1/03 Can’t Cont Prog
	3	INACT 1/03 Death
	4	INACT 1/03Prog Comp Pre End Dt
	WFP	Withdrawal from Program
	ICP	Inability to Continue Program
	DOE	Death of EV
	PCP	Prog Complete Before End Date
	NOS	INAC 5/05 No Show
	CCHG	INAC 5/05 Cancel-Chg of Status
	CHG	INAC 5/05 Change of Status
	DCHG	INAC 5/05 Denied-Chg of Status

GTVSVFT	**SEVIS Drop Below Full Time Reason Code Validation Table**	
	1	Illness/Medical Condition
	2	Difficulty with English
	3	Difficulty with Reading
	4	Not Familiar with U.S.Teaching
	5	Improper Level Placement
	6	Will Complete within Term
	7	Part-Time Commuter Student

GTVSVGO	**SEVIS Governmental Organization Code Validation Table**	
	DOJ	Dept of Justice
	ACT	Action
	AID	Agency For Intl Development
	BBG	Broadcasting Board of Governor
	DOC	Dept of Commerce

GTVSVGO	**SEVIS Governmental Organization Code Validation Table**	
	DOD	Dept of Defense
	DOE	Dept of Energy
	DOED	Dept of Education
	DOI	Dept of Interior
	DOL	Dept of Labor
	DOS	Dept of State
	DOT	Dept of Transportation
	EPA	Environmental Protection Ag
	EXIM	Export-Import Bank
	GAO	General Accounting Agency
	HHS	Health and Human Services
	HMC	Holocaust Memorial Council
	HUD	Housing and Urban Development
	LOC	Library of Congress
	NASA	NASA
	NDH	Nat Endowment for Humanities
	NEA	Nat Endowment for the Arts
	NSF	Nat Science Foundation
	OTHER	Other
	SI	Smithsonian Institution
	TREAS	Dept of Treasury
	USDA	Dept of Agriculture
	USIP	US Institute of Peace
	VA	Dept of Veterans Affairs

GTVSVIO	**SEVIS International Organization Code Validation Table**	
	ECLA	UN Econ Comm. Latin Am/ Carrib
	ECE	UN Economic Commission Europe
	ECA	UN Economic Commission Africa

GTVSVIO	SEVIS International Organization Code Validation Table	
	ECLAC	INAC 5/05 Eco Com Latin Am/ Car
	ECOSOC	UN Economic and Social Council
	EEC	European Economic Community
	ESCAP	UN Econ Comm Asia/Far East
	FAO	UN Food/Agriculture Org
	IAEA	Intl Atomic Energy Agency
	ICAO	Intl Civil Aviation Org
	ILO	Intl Labor Organization
	IMF	Intl Monetary Fund
	IMO	Intl Maritime Organization
	ITU	Intl Telecomm Union
	NATO	North Atlantic Treaty Org
	OAS	Org of American States
	OAU	Org of African Unity
	OECD	Org of Econ Coop. and Develop.
	OTHER	Other
	PAHO	Pan Amer Health Org
	UN	United Nations
	UNCTAD	UN Conf of Trade and Develop
	UNDP	UN Development Program
	UNESCO	UN Ed, Scient and Culture Org
	UNICEF	UN Children’s Fund
	UNIDO	UN Industrial Devel Org
	WB	World Bank
	WHO	World Health Organization
	WMO	World Meteorological Org

GTVSVIT	SEVIS Infraction Type Code Validation Table	
	EXT	Failure to extend DS-2019 in timely manner.

GTVSVIT	**SEVIS Infraction Type Code Validation Table**	
	CON	INAC 5/05 Failure to conclude transfer of program.
	REC	Failure to receive RO/ARO approval before accepting payment
	OTH	Other

GTVSVPC	**SEVIS Position Code Validation Table**	
	110	Central Government Group
	111	Head of Government
	112	Ministerial Level Official
	113	Executive Level Official
	114	Civil Service Employee
	115	Professionals and Scientists
	116	Legislator/Central Government
	117	Judges/Central Government
	118	Manager/State Enterprise
	119	Central Government Other
	120	State, Reg,Prov Govt Group
	121	Governor/Chief of Region
	122	Senior Head of Reg Dept
	123	Exec Level Reg Official
	124	Civil Service/Regional Govt
	125	Prof and Scientist/Regional
	126	Regional Legislator
	127	Regional Judge
	128	Regional Manager
	129	Regional Govt Other
	130	City/Town Government Group
	131	Mayor/City Manager
	132	Head of City Dept
	133	Executive Level City Official
	134	Civil Service/City Govt

GTVSVPC	SEVIS Position Code Validation Table	
	135	Prof and Scientist/City
	136	City Legislator
	137	City Judge
	138	Manager, City Enterprise
	139	City Government Other
	140	International Organization
	141	Head of International Org
	142	Senior Official Intl Org
	143	Intl Org Employee
	210	University Level Group
	211	University President
	212	University Admin Staff
	213	Teaching Staff/University
	214	University Graduate Students
	215	Undergraduate Students/Univ
	216	Medical School Students
	217	Other Professional Students
	218	Post Graduate Medical Trainee
	219	University, Other
	220	Secondary School Group
	221	Secondary School Principal
	222	Secondary School Teacher
	223	Secondary School Student
	229	Secondary School, Other
	230	Elementary School Group
	231	Elementary Principal/Teacher
	239	Elementary School, Other
	240	Special School Group
	241	Special School Head
	242	Special School Teacher
	249	Special School, Other
	310	Private Business Group

GTVSVPC	**SEVIS Position Code Validation Table**	
	311	Private Business Entrepreneur
	312	Corporate Executive
	313	Manager/Private Business
	314	Employee/Private Business
	315	Professional/Scientist, Bus.
	319	Private Business, Other
	320	Self-Employed Group
	321	Self-Employed (Legal)
	322	Self-Employed (Medical)
	323	Self-Employed (Tech)
	329	Self-Employed (Other)
	330	Independent Organization Group
	331	Dir Instit/Corp/Hospital
	332	Mgr-Exec Empl by Instit/Corp
	334	Employee Independent Inst/ Corp
	335	Prof/Scientist Instit/Corp
	339	Independent Org, Other
	340	Agriculture Group
	342	Agricultural Executive
	341	Agricultural Entrepreneur
	343	Agricultural Manager
	344	Agricultural Employee
	345	Agriculture Prof/Scientist
	349	Agriculture, Other
	350	Religion Group
	351	Minister of Religion
	352	Religious Order Member
	353	Theologian
	410	Arts Group
	411	Artist (Graphic Arts)

GTVSVPC	SEVIS Position Code Validation Table	
	412	Author (Playwright,Poet)
	413	Stage/Film Actor
	414	Film/Stage Producer
	415	Composer/Musician
	419	Arts, Other
	420	Sports Group
	421	Athlete
	422	Coach
	429	Sports, Other
	510	Labor Union Group
	512	Labor Union Official
	511	Labor Union Head
	513	Labor Union, Other
	520	Labor Union Ministry Group
	521	Labor Minister or Lab Ag Head
	522	Senior Ministerial Official
	523	Ministerial Employee
	529	Ministry of Labor, Other
	530	Labor Experts Academia Group
	531	Deleted--See 213
	539	Labor Experts Academia, Other
	540	Labor Organization Group
	541	Head of Labor Organization
	542	Labor Organization Employee
	610	Electronic Media Group
	611	Head of TV/Radio Station
	612	Radio/TV Journalist
	613	Electronic Media Technician
	619	Electronic Media, Other
	620	Print Media Group
	621	Editor/Publisher
	622	Journalist

GTVSVPC	SEVIS Position Code Validation Table	
	623	Tech in Print Media Field
	629	Print Media, Other
	630	Film as News Media Group
	631	Film Maker
	639	Film as News Media, Other
	710	Opposition Leader
	720	Opposition Leader (Legislator)
	730	Former Political Official
	790	Important Political Figure
	800	Military
	900	Other

GTVSVTR - SEVIS Termination Reason Code Validation Table		
Code	**Description**	**Usage Indicator**
1	Unauthorized Withdrawal	1
2	Death	1
3	Unauthorized Employment	1
4	Drop Below FT Course of Study	1
5	Full Course Time Exceeded	1
6	Change of Nonimmigrant Status	1
7	Nonimmigrant Stat Chnge Denied	1
8	Expulsion	1
9	Suspension	1
10	Absent from Country for 5 Mos.	1
11	Failure to Enroll	1
12	Costs Exceed Resources	1
13	Transfer Student a No Show	1
14	Denied Transfer	1
15	Extension Denied	1
16	Failing to Maintain Status	1
17	Violation of Change of Status	1

GTVSVTR - SEVIS Termination Reason Code Validation Table		
Code	**Description**	**Usage Indicator**
18	Change of Status Denied	1
19	Change of Status Withdrawn	1
20	Change of Status Approved	1
21	Transfer Withdrawn	1
22	No Show-Manual Termination	1
23	Authorized Early Withdrawal	1
24	No Show-System Termination	1
25	School Withdrawn	1
1	INACT 1/03 Fail to Pursue Prog	2
2	INACT 1/03Fail to Maint Ins	2
3	INACT 1/03 Convict of a Crime	2
4	INACT 1/03 Disciplinary Action	2
5	INACT 1/03 Unauth Employment	2
6	INACT 1/03 Violat Spons Rules	2
7	INACT 1/03 Violating Prog Regs	2
8	INACT 1/03 Fail to Main FT	2
9	INACT 1/03 Involuntary Susp	2
CONVIC	Conviction of a Crime	2
DISCIP	Disciplinary action	2
ENGEMP	Unauthorized employment	2
FALACT	Fail to Pursue EV Prog Activit	2
FALADD	Fail to submit address change	2
FALINS	Fail to maint health Insurance	2
FALSTD	Fail to maint full course	2
INVSUS	Involuntary suspension	2
OTHER	Other	2
VIOEXV	Violating EV program regs	2
VIOSPN	Violating sponsor rules	2

GTVSVTS	Validation Entries for SEVIS Transmittal Status Code Table	
	C	Processing Complete
	P	Pending Response from SEVIS
	N	No action required
	M	Manual - Adjudicated event
	W	Waiting for Batch Transmittal
	X	Not Sent, User Decision
	R	Returned with error

GTVSYSI	System Indicator Validation Table	
	A	Alumni
	G	General
	F	Finance
	R	Financial Aid
	S	Student
	T	Accounts Receivable
	C	Courts
	H	Human Resources
	M	Micro-Faids Interface
	U	Utilities
	N	Position Control
	B	Property Tax
	D	Cash Receipts
	L	Occupational Tax and License
	X	Records Indexing
	IC	Integration Components
	E	Banner XtenderSolutions
	TM	Translation Manager
	FW	Finance Self-Service
	GW	Web General
	VR	Voice Response
	AW	Advancement Self-Service
	SW	Student Self-Service

GTVSYSI	System Indicator Validation Table	
	PW	Employee Self-Service
	LW	Faculty/Advisor Self-Service
	IF	Kiosk (Information Access)
	LC	Luminis Channels for Banner

GUASADM Capture Table	Capture Rule	Capture Columns

GOREMAL		`GOREMAL_EMAIL_ADDRESS`
		`GOREMAL_PREFERRED_IND`
		`GOREMAL_STATUS_IND`

GORIROL		`GORIROL_ROLE` `GORIROL_ROLE_GROUP`

SPBPERS		`SPBPERS_BIRTH_DATE`
		`SPBPERS_LEGAL_NAME`
		`SPBPERS_NAME_PREFIX`
		`SPBPERS_NAME_SUFFIX`
		`SPBPERS_PREF_FIRST_NAME`
		`SPBPERS_SEX SPBPERS_SSN`

SPRADDR		`SPRADDR_ATYP_CODE`
		`SPRADDR_CITY`
		`SPRADDR_CNTY_CODE`
		`SPRADDR_NATN_CODE`
		`SPRADDR_STATUS_IND`
		`SPRADDR_STAT_CODE`
		`SPRADDR_STREET_LINE1`
		`SPRADDR_STREET_LINE2`
		`SPRADDR_STREET_LINE3`
		`SPRADDR_ZIP`

SPRIDEN	`SPRIDEN_CHANGE_IND is NULL`
	
	`SPRIDEN_ENTITY_IND IN (‘P’)`
		`SPRIDEN_CHANGE_IND`
		`SPRIDEN_ENTITY_IND`
		`SPRIDEN_FIRST_NAME`
		`SPRIDEN_LAST_NAME`
		`SPRIDEN_MI`

GUASADM Capture Table	Capture Rule	Capture Columns
SPRTELE		SPRTELE_PHONE_AREA SPRTELE_PHONE_EXT SPRTELE_PHONE_NUMBER

GURTPRF - Toolbar and Menu Preference Table
Toolbar Buttons

```
|69,Workflow Release,wf_release,G
$_WF_BUTTON_PRESSED_TRG;,,414.000,18.000,D,
||70,Workflow Submit,wf_submit,G
$_WF_BUTTON_PRESSED_TRG;,,396.000,18.000,D,
||71,Open Electronic Document,wf_apply,G
$_WF_BUTTON_PRESSED_TRG;,,378.000,18.000,D,
||72,SEM,sem,,,171.000,63.000,E,



 ||73,Banner Help,banner_help,GUAHELP,,144.000,63.000,E,
||74,Internet,internet,,,117.000,63.000,E,
||75,MS Powerpoint,powerpoint,,,99.000,63.000,E,
||76,MS Excel,excel,,
```

Display Horizontal Toolbar	Display Vertical Toolbar	Display Hint	Display Page Name	Display Release Number
Y	Y	Y	Y	Y

Display Database Instance	Display Date and Time	Display Required Item Color	Screen XPosition	Button X Position	Display Page Name
Y	Y	Y	232	224	N

GURUPRF	Personal Preference Table		
Group	**Key**	**String**	**Value**
DATA_EXTRACT	WIN32COMMON	DIRECTORY	c:\temp
REPORT	WEB	DIRECTORY	http://your.report.server/ows-bin/rwcgi60.exe?

GURUPRF	Personal Preference Table		
Group	**Key**	**String**	**Value**
WEBOUTPUT	WEB	DIRECTORY	http://yourserver.com/directory/
MENU	WIN32COMMON	STARTUP_MENU	*MENU
DATA_EXTRACT	WIN32COMMON	MIME_TYPE	FILE
LDAP	AUTHENTICATION	SERVER	ldap://your.ldap.server:port/
LDAP	AUTHENTICATION	DN	DN Name
LDAP	AUTHENTICATION	BIND_USER	Bind user.
LDAP	AUTHENTICATION	BIND_PASSWORD	Bind password.
LDAP	SSL	LOCATION	Wallet Location
LDAP	SSL	PASSWORD	Wallet Password
LDAP	SSL	MODE	Authentication Mode
UI	COLOR	BUTTON	r204g204b153
UI	COLOR	CANVAS	r255g255b255
UI	COLOR	RECORD	r204g204b153
UI	COLOR	SEPARATOR	r204g204b0
UI	COLOR	SCROLLBAR	r204g204b0
UI	COLOR	CODE_PROMPT	r0g0b0

GURUPRF	**Personal Preference Table**		
Group	**Key**	**String**	**Value**
CM	LIST	FORMS	APANAME APAIDEN APAWPRS FOAIDEN FTMAGCY FTMFMGR FTMVEND GXRBANK PPAIDEN RCRSUSP STVINFC SPAIDEN SAAQUIK SRAQUIK SAAEAPS SRIPREL SRQMTCH STVPREL SHAEDIS PEAHIRE PEA1PAY NOAEPAF
UI	ALERT	EXIT	Y
UI	ALERT	CONFIDENTIAL	Y
UI	ALERT	DECEASED	Y
UI	COLOR	MESSAGE_CANVAS	r255g255b255
UI	COLOR	LINKS_CANVAS	r255g255b255
UI	COLOR	TREE_CANVAS	r255g255b255
UI	LINKS	MY_INST	http://www.sungardhe.com/

GURUPRF	**Personal Preference Table**		
Group	**Key**	**String**	**Value**
`UI`	`LINKS`	`MY_LINK_1DESC`	Your first personal link description
`UI`	`LINKS`	`MY_LINK_1EVENT`	Your first personal link URL
`UI`	`LINKS`	`MY_LINK_2DESC`	Your second personal link description
`UI`	`LINKS`	`MY_LINK_2EVENT`	Your second personal link URL
`UI`	`LINKS`	`MY_LINK_3DESC`	Your third personal link description
`UI`	`LINKS`	`MY_LINK_3EVENT`	Your third personal link URL
`UI`	`LINKS`	`MY_LINK_4DESC`	Your fourth personal link description
`UI`	`LINKS`	`MY_LINK_4EVENT`	Your fourth personal link URL
`UI`	`LINKS`	`MY_LINK_5DESC`	Your fifth personal link description
`UI`	`LINKS`	`MY_LINK_5EVENT`	Your fifth personal link URL
`UI`	`LINKS`	`MY_LINK_6DESC`	Your sixth personal link description
`UI`	`LINKS`	`MY_LINK_6EVENT`	Your sixth personal link URL
`IMAGE`	`WEB`	`DIRECTORY`	`c:\YourImageDirectory`
`REPORT`	`WEB`	`SERVICE`	YourServiceName
`HELP`	`WEB`	`DIRECTORY`	`http://your.bannerOH.server/bannerOH/bannerOH`