The eRA/IMPAC II Project

POP TRACK - API Design

Version 1.0

10 October 2001

Prepared For:

Office of Policy for Extramural Research Administration
OD/Office of Extramural Research
National Institutes of Health
Bethesda, Maryland

Prepared by:

Logicon Federal Data/ROW Sciences

Logicon Federal Data/R.O.W. Sciences
1700 research Blvd, Suite 400, Rockville, Maryland 20850-3142.

Table of Contents

1. Overview

Institutes and Centers (IC's) at NIH currently use shadow systems, which provide the ability to define protocols and track target and enrollment counts within the IC. The IC's may wish to continue maintaining this information in their local systems for some period of time, even after the IMPACII Population Tracking module is deployed, for various reasons.

Whatever the case may be, and whatever the reasons, a method of uploading local IC system data to the IMPACII database has to be developed. Since official ORWH reporting will be driven from the IMPACII database, it is crucial that population tracking information entered via a local system is replicated to the enterprise system in a timely manner.

The responsibility for this replication must be shared. The IC must develop software which will extract the information that must be replicated, and send that information to IMPACII, most likely on a nightly basis. It is the IMPACII development team's responsibility to provide an API which will allow IC staff to develop this software. API procedures must be used instead of direct SQL updates, in order to implement the various business rules that must be applied to IMPACII data items.

The API's developed for population tracking will adhere to the standards and conventions which have been adopted for all IMPACII external API's, to this point. These conventions are:

- The procedures are implemented through PL/SQL packages.
- The procedures are as generic as possible to support future expansion.
- The procedures access server-based internal procedures which are *also* used by IMPACII applications.
- All changes made through the procedures are logged in the IMPACII database, for auditing purposes.
- Violations of business rules or failed executions are returned to the calling program through a database exception.

For the Population Tracking module, three new API procedures must be developed. Each API procedure will enforce the business rules that apply to that data when it is entered via the Population Tracking user interface.

2. External and Internal API's

Whenever an API is developed, it is standard practice to develop both an *external* and an *internal* version of that API. This is done so that IMPAC II internal *and* external applications alike can build upon the same, shared set of access methods and business rule logic. The basic design of any external/internal API pair is described in the following paragraphs.

The internal API performs all the database operations and business rule checks that are required to implement the interface. Usually, the internal API parameters and external API parameters will be identical, though this is not always the case. There are occasions where an internal API will accept more column values, due to the broader audience that it must serve. Also, the internal API does not perform any security-layer checking. It is presumed that the caller has already verified the user's IC and package-calling privileges prior to invoking the internal API.

The external API is essentially a *pass-through* to the internal API, but includes logic to enforce the IC and application level security layers which are described in the "Security" section, later on. As mentioned previously, the external API need not expose all the columns that can be manipulated by the internal API. Since the external API serves a specific audience, there will be occasions where the external API does not *need* to collect certain parameters from the calling process. In these cases the external API will hard-code the excluded parameters when calling the internal API.

An internal API will not even be visible to external users of the IMPAC II database. It will only be visible to its equivalent *external* API and to internal IMPAC II applications.

3. IC Extension Application Registration

For tracking usage and to assist IMPAC II administrators in debugging problems, all IMPAC II applications are registered in the IMPAC II database. This same concept also applies to IC extension applications. In the context of IMPACII, IC extension applications fall under the business area with the acronym ICE.

Registering an IC extension application is a two step process:

1. The IC coordinator contacts the IMPACII helpdesk and provides the information necessary to register an extension application. This information will include the application name, software tool (Access, VB, PowerBuilder, etc.), contact person, and a brief description. In the context of IMPACII, the registered name must begin with the ICE acronym. The rest of the registered name (up to 32 characters total) is at the IC's discretion. In order to ensure uniqueness of registered names across IC's IC's may wish to include one of their standard abbreviations within the name. A future version of UserAdmin may allow IC Coordinators to register applications directly without contacting the helpdesk.

2. The IC extension application will be programmed to record its registered name when accessing the IMPACII database. Recording an application name is done by calling the standard PL/SQL procedure dbms_application_info.set_module immediately after logging into IMPACII. This procedure takes two parameters – module_name, which must be set to the registered application name, and action_name, which is the action being performed. In addition, the dbms_application_info_set_action procedure can be used to change the action_name. Use of the action_name is optional in all cases, but may be helpful for debugging purposes to indicate the action an application is currently performing.

It is highly recommended that all IC extension applications be registered. IC extension applications that use the external API *must* be registered. The API will *reject* any transactions received from an application that is not correctly registered. For purposes of the API, this will also improve security, as IC coordinators may elect to keep their registered application names confidential. This will ensure that only IC approved applications are using the API.

4. Security

There will be three levels of security enforced for the external API:

1. Database object security. In order to see and execute an API procedure, a user account must have the ICE_API_ROLE. This role can be granted through the UserAdmin application.
2. IC level security. In order to make assignments or modify data for a specific IC, a user must have privileges to that IC. IC privileges can be granted through UserAdmin. The API procedures will return an exception if the user does not have the correct IC privilege.
3. Application level secur ity. In order for a user to make any changes through an API procedure, they must be using an application that has recorded a valid application name for the current session. For more information on application registration refer to section 2 above.

5. Transaction control

IC extension systems that use the external API will maintain transaction control of all changes. API procedures will never 'commit' database changes. Commits or rollbacks must be made by the calling application. This will allow IC applications to maintain data integrity across multiple procedure calls or in relation to a separate IC database. However it should be noted that this also means IC applications could accidentally maintain outstanding database locks indefinitely on IMPACII objects (e.g.. if an application is programmed to never commit, any records changed by API procedure calls will remain locked until the application exists).

IMPACII administrators will monitor outstanding locks and work with IC developers to resolve any persistent problems. In emergency cases IMPACII administrators will also reserve the right to terminate any sessions that are causing system wide problems.

6. Procedures

As already discussed, *two* API packages will be developed for Population Tracking. There will be one *internal* package that is accessible to the external API and to IMPAC II applications only, and one *external* package that is only accessible to registered IC extension applications. The package names are:

ice_dbms_pop_a_pkg (external)
com_dbms_pop_a_pkg (internal)

As already discussed, the external API procedures will simply enforce the IC and application security layers, and then call the corresponding internal API procedure to apply the appropriate business rules and database operations.

Since the functionality that is provided to the calling process is essentially the same in each case (internal or external), only one description is provided for each API procedure below. However, since differences *do* occasionally exist between the internal and external versions, these will be discussed wherever appropriate.

6.1. protocol_studies_proc

This procedure will be called to insert, update or delete entries in the PROTOCOL_STUDIES_T table.

6.1.1. <u>Parameter List</u>

Trans_type_p INPUT VARCHAR2
Mandatory. Indicates type of transaction requested. Valid values are 'I' for insert, 'U' for update, and 'D' for delete.

Protocol_id_p INPUT/OUTPUT NUMBER
Mandatory. A number that uniquely identifies a protocol or study. For INSERTS, this value is returned to the caller, since this is a system generated identifier. If anything other than a <u>*null*</u> value is INPUT for this parameter on INSERT, the transaction will be rejected. Conversely, for UPDATES and DELETES, this parameter must <u>*not*</u> be null.

Phs_org_code_p INPUT VARCHAR2
Mandatory. Identifies IC for which changes apply.

Protocol_title_p INPUT VARCHAR2
Mandatory. The title of the protocol or study organization.

IC_protocol_id_p INPUT VARCHAR2
Optional. An alphanumeric code designated by the IC to help identify a protocol.

Clinical_trial_phase_code_p INPUT VARCHAR2
Optional. A code that indicates whether the protocol or study is NIH Defined Phase III.

Phase_3_code_p INPUT VARCHAR2
Optional. A code that indicates whether the protocol or study is NIH Defined Phase III.

Protocol_status_code_p INPUT VARCHAR2
Optional. A code that indicates the status of a protocol or study.

Study_cmnt_p INPUT VARCHAR2
Optional. Comments about the protocol or study.

study_end_date_p INPUT VARCHAR2
Optional. The date the protocol or study ends.

study_start_date_p INPUT VARCHAR2
Optional. The date the protocol or study starts.

tracking_exception_code_p INPUT VARCHAR2
Optional. A code that indicates whether the protocol or study is excepted from tracking.

6.1.2. Business Rules

There are no business rules that the API must apply for protocol_studies_t.

6.1.3. Usage Notes

The IC upload software that drives this process should ensure that any new protocols are inserted *before* they are referenced in appl_protocols_t.

6.2. *appl_protocols_proc*

This procedure will be called to insert, update, or delete entries in the APPL_PROTOCOLS_T table.

6.2.1. Parameter List

Trans_type_p INPUT VARCHAR2
Mandatory. Indicates type of transaction requested. Valid values are 'I' for insert, 'U' for update, and ‘D’ for delete.

Appl_protocol_id_p INPUT/OUTPUT NUMBER
Mandatory. A unique number that identifies a relationship between an application and a protocol or study. If anything other than a *null* value is INPUT for this parameter on INSERT, the transaction will be rejected. Conversely, for UPDATES and DELETES, this parameter must *not* be null.

Phs_org_code_p INPUT VARCHAR2
Mandatory. A code that uniquely identifies the PHS organization that administers the protocol.

Appl_id_p INPUT NUMBER

Mandatory. A unique number that identifies application.

Protocol_id_p INPUT NUMBER
Mandatory. A number that uniquely identifies a protocol or study.

Interim_final_code_p INPUT VARCHAR2
Mandatory. A code that indicates whether the subject counts are interim or final. The final code is used at project close out or at the end of a competing segment of a grant.

Enrollment_approval_code_p INPUT VARCHAR2
Optional. A flag that indicates whether the subject counts have been approved by an authorized individual.

Target_approval_code_p INPUT VARCHAR2
Optional. A flag that indicates whether the target counts have been approved by an authorized individual.

Country_seq_num_p INPUT NUMBER
A number which uniquely identifies a country.

Enrollment_approved_by_id_p INPUT VARCHAR2
The IMPACII user ID of the person approving the enrollment data.

Enrollment_cmnt_p INPUT VARCHAR2
Comments about the enrollment data.

Enrollment_exception_code_p INPUT VARCHAR2
A code that identifies a reason why enrollment data does not exist for a given year.

Enrollment_status_code_p INPUT VARCHAR2
A code that identifies the status of the enrollment of a protocol or study.

Foreign_domestic_code_p INPUT VARCHAR2
A code that identifies if the population counts contain foreign subjects.

Performance_period_fy_p INPUT NUMBER
The fiscal year in which the work was performed.

Report_submitted_fy_p INPUT NUMBER
The fiscal year of the progress report or competing segment of an application.

Justification_text_p INPUT VARCHAR2
This field is used to store a justification on why subject counts were changed after the final report was created.

Target_approved_by_id_p INPUT VARCHAR2

The IMPACII user ID of the person approving the target data.

Target_cmnt_p INPUT VARCHAR2
Comments about target data.

Target_exception_code_p INPUT VARCHAR2
A code that identifies a reason why target data does not exist for a given year.

Tracking_exception_code_p INPUT VARCHAR2
A code that identifies those applications that are excepted from tracking.

6.2.2. Business Rules

1. Only the grants that must be tracked need to be inserted. The human_subject_code for the application/subproject indicates that human subjects are involved: appls_t.human_subject_code not in (‘10’, ’98’, ’99’). The tracking_exception_code for the application/subproject indicates that tracking is required: appls_t.tracking_exception_code = ‘00’.

2. Protocols are not shared across IC’s. Only a protocol’s administering IC can assign that protocol to a grant application. Therefore, *phs_org_code_p* must equal *protocol_studies_t.phs_org_code*, for the protocol being assigned.

3. Only the administering IC for a grant application can assign protocols to it Therefore, *phs_org_code_p* must equal *appls_t.admin_phs_org_code*, for the application being assigned.

4. If the protocol status is closed (protocol_studies_t.protocol_status_code = ‘C’), the protocol *cannot* be assigned to any grant.

6.2.3. Usage Notes

The IC upload software that drives this process should ensure that any new appl protocols are inserted *before* they are referenced in subject_counts_t.

6.3. *subject_counts_proc*

This procedure will be called to insert, update, or delete entries in the SUBJECT_COUNTS_T table.

6.3.1. Parameter List

Trans_type_p INPUT VARCHAR2
Mandatory. Indicates type of transaction requested. Valid values are 'I' for insert and 'U' for update and ‘D’ for delete.

Phs_org_code_p INPUT VARCHAR2

Mandatory. A code that uniquely identifies the PHS organization that administers the protocol.

Appl_protocol_id_p INPUT NUMBER
Mandatory. A unique number that identifies a relationship between an application and a protocol or study.

Count_type_p INPUT VARCHAR2
Mandatory. A code that identifies whether the count is for target or enrollment.

Hispanic_indicator_code_p INPUT VARCHAR2
Mandatory. A code that identifies whether the subject counts involved in the study represent the Hispanic ethnic group.

Gender_code_p INPUT VARCHAR2
Mandatory. Type of genders involved in the study.

Race_type_code_p INPUT NUMBER
Mandatory. A racial ethnic type code associated with a person.

Subject_count_num_p INPUT NUMBER
Optional. The number of subjects involved in the study.

Please note, the following parameter will apply to the ***INTERNAL API ONLY***:

Race_sub_type_code_p INPUT NUMBER
Mandatory. Racial ethnic subtypes for each type code within the racial_ethnic_type_t table. Wherever this value is not collected, is not known, or does not apply, a value of “0” (indicating not applicable) should be provided. At this time, the external API will *always* send a value of “0” for this parameter, since there are currently no requirements to designate subject counts at this level.

6.3.2. Business Rules

1. Only the administering IC for a grant application can assign subject counts to it Therefore, *phs_org_code_p* must equal *appls_t.admin_phs_org_code*, for the application in question.

2. If the protocol status is closed (protocol_studies_t.protocol_status_code = ‘C’), any type of transaction *will be allowed* on subject_counts. It is assumed that even though a protocol may be closed, there will still be a need to correct or complete data entry for the subject counts on any given grant.

3. For the same reasons, enrollment counts will still be accessible through the API, even when the enrollment status is closed. It will be difficult at an API level to determine whether to allow or disallow these operations on enrollment counts, since data entry and correction within the system may still be ongoing, even after the official enrollment period has ended.

4. The hispanic_indicator_code will be checked on insert, to ensure that there is no mixture of “old form” (hispanic indicator code : ‘O’) values and “new form” (hispanic indicator code : ‘H’, ‘T’ or ‘U’) values on the same appl protocol.

6.3.3. Usage Notes

The IC upload software that drives this process should ensure that any new protocol or appl protocol records are inserted, before inserting subject_counts_t records that may reference them.