New Software for Ensemble Creation in the SSC Operations Database

Russ Laher (laher@ipac.caltech.edu) and John Rector (jar@ipac.caltech.edu)
Spitzer Science Center (SSC), California Institute of Technology, Pasadena CA

July 20, 2004

Table of Contents
Introduction......................................................................................................................... 1
Background......................................................................................................................... 2
Summary of New and Improved Features of Ensemble-Creation Software....................... 3
Ensemble-Creation Rules.................................................................................................... 3
Special Considerations for Ensemble-Creation Rules........................................................ 3
New Database Schema for Ensemble Creation ................................................................. 6
New Database Stored Procedures/Functions for Ensemble Creation................................. 8
New Perl Script for Ensemble Creation.............................................................................. 9
Software Usage .................................................................................................................11
Appendix A: “On-Line” Tutorial for createEnsembles.pl ................................................ 13
Appendix B: Data Contents of the ensRules and ensPlScripts Database Tables.............. 15
Appendix C: Test-Mode Sample Output from createEnsembles.pl.................................. 36
Appendix D: Normal-Mode Sample Output from createEnsembles.pl............................ 38

Introduction

Many SSC downlink-data pipelines for computer processing Spitzer-Space-Telescope DCEs (data-collection events, which are raw images) rely on explicit groupings, or ensembles, of images made a priori in the SSC operations database for en masse data-processing of images. Some Basic-Calibrated-Data (BCD) pipelines, most calibration pipelines for creating reduced-noise calibration data, and most post-BCD pipelines for creating co-added-image mosaics, band-merged source lists, and other high-level science products all require pre-defined ensembles of images. Information about how the DCEs are grouped together as ensembles is stored in the SSC operations database so that a simple database query can retrieve the required pipeline input images.

How this ensemble information is populated in the database is central to this report. This is routinely done by the pipeline operator after a new set of DCE metadata has been created in the database. Initially, a “quick-and-dirty” perl script was developed by SSC pipeline operator Ron Beck to perform this task, and this was used successfully in operations for the first eight months following the Spitzer’s launch. It quickly became apparent that the script would be more convenient if it were faster (some campaigns took as long 12 hours to make all the required ensembles). Furthermore, because the rules for ensemble creation were, for a long time, evolving, which necessitated a change to the script each time (with an iteration through the Change-Control Board!), a table-driven
design for specifying ensemble-creation rules was identified as being an important element of the next version of ensemble-creation software.

The purpose of this report is to describe this next version of new and improved software for ensemble creation. The top-level portion of the software is still written in perl, but there are new database tables and database stored procedures/functions that are used by the software to accomplish the two main objectives of faster ensemble creation and table-driven ensemble-creation rules.

**Background**

The SSC operations database, which is an Informix database, includes three tables called **ensembles**, **dceSets**, and **ensembleSets**, which are for storing information about how ensembles of images are defined and how they are to be processed. A UML diagram showing how these database tables are interrelated is given in Figure 1.

![Figure 1. The ensembles, dceSets, and ensembleSets database tables.](image)

The **dceSets** database table makes one-to-many associations between a dceSetId (a database ID number for a set of DCEs) and multiple dceIds (database serial IDs for DCEs). A collection of dceIds for a given dceSetId define what is called a DCE set. The **ensembles** database table makes one-to-many associations between a DCE set and the ensemble-processing pipelines that process the DCE set. Each record in the **ensembles** database table is indexed by a unique ensId, which is a database serial ID number for an ensemble. Each pipeline is indexed by a unique plScriptId, which is a database ID number for a pipeline. In most cases, there is just one ensemble-processing pipeline associated with a DCE set, so that just one ensembles record is needed. In some cases, there is more than one ensemble-processing pipeline associated with a DCE set, and so multiple ensembles records are needed for associating with a given dceSetId. Finally, the **ensembleSets** database table allows for “ensembles of ensembles” to be pipeline-processed *en masse*. An “ensemble of ensembles” consists of two or more ensembles records, each with an “in” ensId, that are associated together via an “out” ensId, which corresponds to another ensembles record with a null value for dceSetId and a plScriptId of the “ensemble of ensembles” pipeline. The null dceSetId value is an indicator that the ensemble is an “ensemble of ensembles”.

---

1. design for specifying ensemble-creation rules was identified as being an important element of the next version of ensemble-creation software.

2. The purpose of this report is to describe this next version of new and improved software for ensemble creation. The top-level portion of the software is still written in perl, but there are new database tables and database stored procedures/functions that are used by the software to accomplish the two main objectives of faster ensemble creation and table-driven ensemble-creation rules.

3. **Background**

   The SSC operations database, which is an Informix database, includes three tables called **ensembles**, **dceSets**, and **ensembleSets**, which are for storing information about how ensembles of images are defined and how they are to be processed. A UML diagram showing how these database tables are interrelated is given in Figure 1.

![Figure 1. The ensembles, dceSets, and ensembleSets database tables.](image)

4. The **dceSets** database table makes one-to-many associations between a dceSetId (a database ID number for a set of DCEs) and multiple dceIds (database serial IDs for DCEs). A collection of dceIds for a given dceSetId define what is called a DCE set. The **ensembles** database table makes one-to-many associations between a DCE set and the ensemble-processing pipelines that process the DCE set. Each record in the **ensembles** database table is indexed by a unique ensId, which is a database serial ID number for an ensemble. Each pipeline is indexed by a unique plScriptId, which is a database ID number for a pipeline. In most cases, there is just one ensemble-processing pipeline associated with a DCE set, so that just one ensembles record is needed. In some cases, there is more than one ensemble-processing pipeline associated with a DCE set, and so multiple ensembles records are needed for associating with a given dceSetId. Finally, the **ensembleSets** database table allows for “ensembles of ensembles” to be pipeline-processed *en masse*. An “ensemble of ensembles” consists of two or more ensembles records, each with an “in” ensId, that are associated together via an “out” ensId, which corresponds to another ensembles record with a null value for dceSetId and a plScriptId of the “ensemble of ensembles” pipeline. The null dceSetId value is an indicator that the ensemble is an “ensemble of ensembles”.

---

2
This database method of defining ensembles of images, which was developed by SSC software engineer John Rector, is generic and independent of which of the three Spitzer-Space-Telescope science instruments that acquired the images (IRAC, MIPS, or IRS). This is because database IDs for DCEs, sets of DCEs, and corresponding pipeline-dependent ensembles are used to make the actual associations, and these database IDs are independent of instrument. This database schema captures all of the use-cases covered by all ensemble-processing at the SSC in the simplest possible way.

Summary of New and Improved Features of Ensemble-Creation Software

1. Database-table-driven ensemble-creation rule specification
2. Faster performance
3. Test mode for checking ensemble-creation rule specification
4. Post-mortem database debugging capability (in both test and normal modes)
5. Useful diagnostic outputs from software
6. Ensemble-creation rule-tracking in \textbf{ensembles} database table
7. Minimum number of DCEs in ensemble specified with ensemble-creation rule
8. Special “X+100” ensemble-plScriptId logic for MIPS:Ge pipelines
9. “On-line” tutorial on the use of the software (see Appendix A)

Ensemble-Creation Rules

For the present design, on which the authors collaborated in the early phases of the effort, the ensemble-creation rules are stored in two new database tables, called \textbf{ensRules} and \textbf{ensPlScripts}. Records in the \textbf{ensRules} database table specify how various attributes of DCEs are used both to query the database for specific DCEs and as discriminants for grouping the returned DCEs into one or more DCE sets. Each \textbf{ensRules} record is indexed by ruleId (a database ID that is unique for each ensemble-creation rule). Records in the \textbf{ensPlScripts} database table specify the ensemble-processing pipelines that are to be associated with each ruleId, in order to complete the specification of information needed to create ensembles. This design results in the economy of allowing a given DCE set to be associated with more than one ensemble-processing pipeline. A UML diagram showing how these tables are interrelated is given in Figure 2.

The table-driven feature of the design is encapsulated in the “sql” field of the \textbf{ensRules} database table, which stores the actual SQL query for the DCEs and their unique sets of discriminants for creating ensembles. For example, one of the rules for IRAC darkcal ensembles could be explained in English by the following two sentences:

1. For a given campaign request, get all the DCEs with initPlScriptId=1, requests.reqtypename='AOR', iracMap.highDynamic='f', and exposures.cycleNum=NULL
2. Group these DCEs in groups where each group has the same chanNum, fowlerNum, waitPeriod, and dceNum.
The filtering attributes that are listed in the first sentence refer to several database tables, including \texttt{dces} (implicitly associated with the \texttt{initPlScriptId} field), \texttt{requests}, \texttt{iracMap}, \texttt{exposures} database tables. The grouping attributes that are listed in the second sentence are the discriminants for creating ensembles. The discriminants for all ensemble-creation rules specified thus far at the SSC, along with their database data-types, are listed as follows:

\begin{verbatim}
dces.initPlScriptId: smallint
dces.chanNum: smallint
dces.exposureNum: smallint
dces.fowlerNum: smallint
dces.waitPeriod: integer
dces.dceNum: smallint
dces.primaryField: smallint
exposures.cycleNum: smallint
exposures.aperture: smallint
exposures.clusterPosNum: smallint
exposures.frameNum: smallint
targetFixedCluster.arraycoord: boolean
\end{verbatim}

These discriminants are “hard-wired” into the new and improved software, in order to satisfy the faster ensemble-creation requirement. New rules can be added to the \texttt{ensRules} and \texttt{ensPlScripts} database tables via simple SQL insert statements, thus satisfying the “table-driven” requirement. However, any new rules to be added are limited to using only the discriminants listed above. Modifications to the software, database schema, and database stored procedures are required if additional discriminants are needed.

The “sql” field of the \texttt{ensRules} database table for the above example is populated with the following SQL statement:

\begin{verbatim}
insert into EnsTempList
    (initPlScriptId, chanNum, fowlerNum, waitPeriod, dceNum)
select unique
    a.initPlScriptId, a.chanNum, a.fowlerNum, a.waitPeriod, a.dceNum
\end{verbatim}
from dces a, exposures b, requests c, iracMap d
where a.reqkey = b.reqkey
and a.exposureNum = b.exposureNum
and b.reqkey = c.reqKey
and c.reqkey = d.reqKey
and b.cycleNum is NULL
and c.reqTypeName = 'AOR'
and d.highDynamic = 'f'
and a.initPlScriptId = 1
and a.reqKey = reqKey_

Notice that, although the grouping attributes (discriminants) are limited to the above list, any of the fields in any of the tables in the database can be used as filtering attributes (no limitations here). It may be prudent to enact a policy that limits the number of tables joined in an ensemble-creation rule to, say, five, in order to prevent runaway database queries.

Upon examination of this complex SQL statement, it goes without saying that it could not have been constructed without expert knowledge of database SQL and familiarity with the SSC operations database schema. The price of this complexity, however, is control with surgical precision over which DCEs are to be included in, as well as excluded from, the ensembles of interest.

Other fields in the ensRules database table that require further explanation are as follows. The “make” field is a Boolean field that indicates whether the rule is active. Over time, rules will evolve and some will be replaced or deprecated; in this case make=’f’ will be set manually (there is no default value). The “ensOfEns” field is a Boolean field to flag those rules that correspond to “ensembles of ensembles” rather than simple ensembles (default value is ‘f’). The “minInputs” field gives the minimum number of scheduled DCEs for the ensemble (default value is 2); if the number of scheduled DCEs is less than the minInputs value, then the ensembles, dceSets, and ensembleSets database records, as appropriate, will not be made.

A new “ruleId” field has been added to the ensembles database table, for tracking purposes. Thus, it now becomes possible to associate an ensemble with the rule that created it. This feature is in itself a distinct improvement. Using this feature, it now becomes easy to analyze how a particular ensemble was created. A database referential constraint will not be created to assure that when this ruleId field is populated (null values are allowed to accommodate existing ensembles records, which will not be back-filled), there exists a corresponding record in the ensRules database table. The following external controls and procedures will be implemented instead: 1) Deleting or updating ensRules and ensPlScripts database records is not allowed, but they can be made obsolete by setting “make”=‘f’; and 2) new ensRules and ensPlScripts database records may be added for new or replacement ensemble-creation rules.

Appendix B gives a complete listing of all the ensemble-creation rules used for ensemble-processing pipelines at the SSC. The information given in this appendix is broken down by the various fields populated in the ensRules and ensPlScripts database tables.
Special Considerations for Ensemble-Creation Rules

When the plScriptId field is set to -100 in the ensPlScripts database table, special logic is triggered for the associated ensemble-creation rule to specially compute the plScriptId for the ensemble by adding the value 100 to the initPlScriptId value. This feature is used only by MIPS:Ge pipelines (e.g., ruleIds 300, 310, 320, 325, 340, and 343), in conformance with the special allocation of plScriptIds that was set up for these pipelines. It follows that initPlScriptId must be specified as a discriminant in the ensemble-creation rule for rules that invoke this special logic. In fact, the special logic will not be invoked unless initPlScriptId>0, which means internally that it was explicitly listed as an ensemble-creation rule discriminant for the rule of interest.

Many of the rules in Appendix B require the use of the database function nvl() to assure that any null values for ensemble-creation discriminants are changed to the value -1 to facilitate computer-coding the core ensemble-creation processes into database stored functions. Null values for the discriminants are not allowed by the new ensemble-creation database schema and stored procedures/functions to be discussed in the following sections.

The ensemble-creation discriminant called “arraycoord” is a Boolean-datatype field in the SSC operations database-table targetFixedCluster. However, it is required that arraycoord Boolean values be converted to 0/1 values when processed within the framework of the new ensemble-creation database schema and stored procedures/functions (see next two sections).

All ensemble-creation rules involving a channel number greater than or less than a certain value should be constructed in such a way as to bracket the desired channel number values, in order to avoid the value -1 (which is the out-of-domain value in the new ensemble-creation schema – see next section). The same guideline also applies for excluding bogus channel numbers, such as channel number 10 in the case of IRAC resets/aborts (for which no DCEs are acquired).

New Database Schema for Ensemble Creation

Figure 3 shows a UML diagram of all database tables used for ensemble creation, including eight new temporary tables (actually, these are not temporary tables that are created and dropped dynamically by database stored procedures, but rather it is the data content of these tables that is temporary). The data contents of the temporary tables, which normally contain information about the ensembles created for some previously specified campaign request, are completely deleted prior to populating the temporary tables with ensemble-creation information for another campaign request.
Figure 3. New database schema for ensemble creation. The ensemble-creation “temporary” tables are highlighted in yellow.
The diagram shows how the temporary tables tie into the new **ensRules** and **ensPlScripts** database tables, as well as the pre-existing **ensembles**, **dceSets**, and **ensembleSets** database tables.

The discriminants for ensemble creation, which were listed in the previous section, are explicit fields in the **ensTempList**, **ensTempList2**, **ensOfEnsTempList** and **ensOfEnsTempList2** database tables (highlighted in blue font in Figure 3). The schema for these tables must be augmented with any new discriminants that may be needed in the future. The discriminants are set to the value -1 by default unless explicitly set when the records are created (null values are not allowed).

Because these eight temporary tables are temporary, they can be dropped from the database, modified via schema changes (for example, to add new ensemble-creation discriminants), and re-created during the quiescent times between episodes of ensemble creation.

**New Database Stored Procedures/Functions for Ensemble Creation**

Several new database stored procedures and functions have been developed for interacting with the ensemble-creation database tables directly (see Table 1). Database stored procedures and function are pre-compiled and, therefore, give much faster performance than standard SQL statements. This not only helps to satisfy the faster ensemble-creation requirement, but it is also in conformance with the SSC’s policy of requiring that interactions with the SSC operations database be done via stored procedures (unless exceptions are made). The database stored procedures are written in Informix stored-procedure language (SPL).

<table>
<thead>
<tr>
<th>Database stored procedure/function</th>
<th>Return value(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>getEnsRules()</td>
<td>Table with columns ruleId, instrument, sql, make, ensOfEns, minInputs, and comment</td>
</tr>
<tr>
<td>getEnsPlScripts()</td>
<td>Table with columns ruleId and plScriptId</td>
</tr>
<tr>
<td>getReqMode(reqKey)</td>
<td>Requests.reqMode</td>
</tr>
<tr>
<td>deleteAllEnsTempLists</td>
<td>None</td>
</tr>
<tr>
<td>getEnsGroupsFromEnsTempList(ruleId)</td>
<td>Table with columns initPlScriptId, chanNum, cycleNum, aperture, frameNum, primaryField, arraycoord, dceSetId, expectedInputs, and (representative) dceId</td>
</tr>
<tr>
<td>getEnsSetsFromEnsOfEnsTempList3(ruleId)</td>
<td>Table with columns outEnsId and inEnsId</td>
</tr>
<tr>
<td>createEnsembles(ruleId, test)</td>
<td>ruleId, ensId, ensPlScriptId, (representative) dceId, dceSetId, and expectedInputs</td>
</tr>
<tr>
<td>createEnsembleSets(ruleId, test)</td>
<td>ruleId, ensId, ensPlScriptId, (representative) dceId, dceSetId (this is always null), and expectedInputs</td>
</tr>
</tbody>
</table>
Stored functions createEnsembles and createEnsembleSets do most of the work and are the most complex in terms of SPL programming. These functions include a test input-flag. When this flag is set to 1, only the ensemble-creation temporary database tables are loaded with records for diagnostic purposes; if this flag is set to 0, then not only the ensemble-creation temporary database tables, but also the ensembles, dceSets, and ensembleSets database tables, as appropriate, are loaded with records. Note that these functions cannot be executed as stand-alone processes under dbaccess (Informix program for interacting with a database), unless the correct ensemble-creation temporary tables are first pre-loaded.

The database stored procedures/functions in Table 1 are called by the ensemble-creation software via perl API functions of the same name, which also have been developed as part this effort. The perl-API functions are called by the perl script described in the next section.

**New Perl Script for Ensemble Creation**

A new perl script called createEnsembles.pl has been implemented based on the design described in detail above. It is this script that is executed by the user, in order to create ensembles for a given campaign request. Figure 4 gives a flow chart of the steps performed by the perl script.

The perl script can be used to do either or both of the following tasks:
1. Perform ensemble creation (test or normal modes).
2. List ensemble-creation rules currently stored in the database.

The basic command-line input to the perl script when it is used to create ensembles is the campaign request number (-r reqKey option). Although not indicated in the flow chart, the perl script queries the database in order to ascertain the Spitzer science instrument associated with the campaign request. Based on this information, only those ensemble-creation rules that are associated with the corresponding instrument are invoked. This feature also contributes to the aforementioned faster ensemble-creation requirement.

The SQL statements for the ensemble-creation rules that are stored in the ensRules database table are executed in the database via system calls to dbaccess. This is necessary because the SQL statements are “freely” formulated and, therefore, cannot be executed via database stored procedures/functions (which must be custom developed and loaded into the database by a DBA). Because of the flexibility needed for constructing table-driven ensemble-creation rules, a waiver to the SSC’s policy of using only stored procedures/functions to interact with the database is a requirement of the software.

A command line flag is available to do ensemble creation in the test mode (-t switch). The test mode loads the ensemble-creation temporary database tables, but does not make any ensembles, dceSets, or ensembleSets database records. The perl script will output a diagnostic table when executed in the test mode, just as in the normal mode. The
difference between the two modes is that, in the test mode, the ensId values will be set to artificially negative values as a test-mode indicator, but nevertheless these values will be unique for a given campaign request. EnsId values less than or equal to -1,000,000 are reserved for indicating outEnsIds for ensembles of ensembles. The diagnostic outputs

![Flow chart for the perl script createEnsembles.pl.](image-url)

Figure 4. Flow chart for the perl script createEnsembles.pl.
from the perl script allow the user to determine whether the specified ensemble-creation rules (as specified by the ensRules and ensPlScripts database tables) create the ensembles as intended. The ensemble-creation temporary database tables are also loaded with negative ensId values in the test mode, so that querying these tables gives additional and more detailed post-mortem diagnostics. The previous section gives additional pertinent details of how the test flag interfaces into the ensemble-creation database stored functions.

When the perl script is used to list the ensemble-creation rules, the -l option is invoked. With this option, either IRAC, MIPS, IRSX, or ALL (case sensitive) should be specified, depending on which ensemble-creation rules are desired.

The database utility library file SODB_Utils.pl, as well as SDM packages sdm_dbmsAccess.pm and sdm_sodbDnl.pm, are required by createEnsembles.pl. These libraries provide the interfaces for the database interactions that use stored procedures and functions. Several new wrapper functions and perl-API functions have been developed and added to SODB_Utils.pl and sdm_sodbDnl.pm, respectively, as part of this effort.

Appendix C lists a sample output file created by createEnsembles.pl in the test mode, and Appendix D lists a sample output file created by createEnsembles.pl in the normal mode (for a different campaign request).

**Software Usage**

Only one instance of createEnsembles.pl is to and should be executed at one time. No interlocks have been implemented to prevent one instance from stepping on another. Furthermore, since the current data contents of the ensemble-creation temporary tables may be under investigation, especially after new ensemble-creation rules have just been added, some coordination is required before starting the next instance.

There are five environment variables that must be set prior to executing the perl script (see Table 2).

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERL_PATH</td>
<td>Path to perl interpreter</td>
</tr>
<tr>
<td>SODB_ROLE</td>
<td>Database role; e.g., “pipelineOpsRole”</td>
</tr>
<tr>
<td>TARGETDB</td>
<td>Target database; e.g., “sodb_dnl5”</td>
</tr>
<tr>
<td>INFORMIXSERVER</td>
<td>Database server name; e.g., “sodb1”</td>
</tr>
<tr>
<td>SIRTF_SCRIPTS</td>
<td>Path to SODB utility library subroutines.</td>
</tr>
</tbody>
</table>
Additionally, the PERL5LIB environment variable must include the path to the SDM perl-API packages (see previous section).

An “on-line” tutorial that lists the command line options and switches, as well as sample command-lines, is outputted when createEnsembles.pl is executed without any command-line options or switches. (environment variable PERL_PATH must be set first.) Appendix A lists the tutorial.
Appendix A: “On-Line” Tutorial for createEnsembles.pl

createEnsembles.pl, Version 1.0

By Russ Laher (laher@ipac.caltech.edu)
Copyright (C) 2004
Spitzer-Space-Telescope Science Center (SSC),
California Institute of Technology, Pasadena, CA

For a given request, creates the ensembles, dceSets, and ensembleSets records specified by the information in the database ensRules and ensPlScripts tables. Optionally lists the ensemble-creation rules, either for a given instrument or for all instruments. A test mode is available to verify the correctness of the ensemble-creation rules without actually creating the ensembles, dceSets, and ensembleSets records.

Command-line options (required unless otherwise stated):

-r <reqKey> Request key for which to create ensembles; optional. If this option is not specified, then neither will ensembles be created nor will the ensemble-creation temporary tables be pre-loaded.
-o <filename> Output filename; required.
-l <instrument> Generate list of ensemble-creation rules for either IRAC, MIPS, IRSX, or ALL; optional. Default is no ensemble-creation rules are listed if this option is not specified.

Command-line switches:

-t Test pre-load the ensemble-creation temporary tables only. This is a test mode, and is applicable only when the -r <reqKey> option is specified. It is useful for verifying that the desired ensembles will be created without actually creating them. The output file has ensId < 0 to indicate that the ensemble was not created.
-v Verbose switch

Environmental variables (no default values set):

SODB_ROLE Database role. E.g., "pipelineOpsRole". Required.
TARGETDB Target database. E.g., "sodblink2". Required.
INFORMIXSERVER Name of database server. E.g., "sodbl1". Required.
SIRTF_SCRIPTS Path to database utility subroutines; required.

Sample command-lines:

To get listing of all ensemble-creation rules:

createEnsembles.pl -l ALL -o RulesForAllInstruments.out

To get listing of IRS ensemble-creation rules:

createEnsembles.pl -l IRSX -o RulesForIrs.out
To load ensemble-creation temporary tables for a given request as a test (without actually creating any Ensembles, dceSets, and EnsembleSets records):

    createEnsembles.pl -o req9426688.out -r 9426688 -t

To create any Ensembles, dceSets, and EnsembleSets records (and also preload the ensemble-creation temporary tables):

    createEnsembles.pl -o req9426688.out -r 9426688
Appendix B: Data Contents of the ensRules and ensPlScripts Database Tables

The data content of the ensRules and ensPlScripts database tables can be obtained anytime via the -l option of createEnsembles.pl:

%% createEnsembles.pl –o allRules.txt –l ALL
%% cat allRules.txt

createEnsembles.pl, Version 1.0

By Russ Laher (laher@ipac.caltech.edu)
Copyright (C) 2004
Spitzer-Space-Telescope Science Center (SSC),
California Institute of Technology, Pasadena, CA

For a given request, creates the ensembles, dceSets, and ensembleSets records specified by the information in the database ensRules and ensPlScripts tables. Optionally lists the ensemble-creation rules, either for a given instrument or for all instruments. A test mode is available to verify the correctness of the ensemble-creation rules without actually creating the ensembles, dceSets, and ensembleSets records.

Database, role = sodb_dnl5, dnlDevRole

Date/time = Fri May  7 07:42:35 2004

Ensemble-creation rules for ALL instruments (from database EnsRules and EnsPlScripts tables):

```
ruleId    = 1
instrument = IRSX
sql       = insert into EnsTempList
            (initPlScriptId, chanNum, aperture)
            select unique
            a.initPlScriptId, a.chanNum, b.aperture
            from dces a, exposures b
            where a.reqkey = b.reqkey
            and a.exposureNum = b.exposureNum
            and a.initPlScriptId = 2000
            and a.reqKey = reqKey_;
make      = t
ensOfEns  = f
minInputs = 1
comment   = IRSX drk. For all DCEs with plscriptId=2000 (IRS pre-processing pipeline for darks) in a given request, form DCE groups with the same chanNum and aperture.
plScriptId(s) = 2001

ruleId    = 2
instrument = IRSX
sql       = insert into EnsTempList
            (initPlScriptId, chanNum, exposureNum)
            select unique
```

15
a.initPlScriptId, a.chanNum, b.exposureNum
from dces a, exposures b
where a.reqkey = b.reqkey
and a.exposureNum = b.exposureNum
and a.initPlScriptId = 2005
and a.reqKey = reqKey_;  
make = t
ensOfEns = f
minInputs = 1
comment = IRSX spl.
plScriptId(s) = 2006
ruleId = 3
instrument = IRSX
sql = insert into EnsTempList
(initPlScriptId, chanNum, exposureNum)
select unique
a.initPlScriptId, a.chanNum, b.exposureNum
from dces a, exposures b
where a.reqkey = b.reqkey
and a.exposureNum = b.exposureNum
and a.initPlScriptId = 2010
and a.reqKey = reqKey_;  
make = t
ensOfEns = f
minInputs = 1
comment = IRSX spf.
plScriptId(s) = 2011
ruleId = 4
instrument = IRSX
sql = insert into EnsTempList
(initPlScriptId, chanNum, exposureNum, aperture)
select unique
a.initPlScriptId, a.chanNum, b.exposureNum, b.aperture
from dces a, exposures b
where a.reqkey = b.reqkey
and a.exposureNum = b.exposureNum
and a.initPlScriptId = 2015
and a.reqKey = reqKey_;  
make = t
ensOfEns = f
minInputs = 1
comment = IRSX coad and bs.
plScriptId(s) = 2016, 2025
ruleId = 100
instrument = IRAC
sql = insert into EnsTempList
(initPlScriptId, chanNum, fowlerNum, waitPeriod, dceNum)
select unique
a.initPlScriptId, a.chanNum, a.fowlerNum, a.waitPeriod, a.dceNum
from dces a, exposures b, requests c, iracMap d
where a.reqkey = b.reqkey
and a.exposureNum = b.exposureNum
and b.reqkey = c.reqKey
and c.reqkey = d.reqKey
and b.cycleNum is NULL
and c.reqTypeName = 'AOR'
and d.highDynamic = 'f'
and a.initPlScriptId = 1
and a.reqKey = reqKey;

make = t
ensOfEns = f
minInputs = 2
comment = IRAC AOR drk.
plScriptId(s) = 2

ruleId = 101
instrument = IRAC
sql = insert into EnsTempList
     (initPlScriptId, chanNum, fowlerNum, waitPeriod,
      dceNum)
     select unique
      a.initPlScriptId, a.chanNum, a.fowlerNum, a.waitPeriod,
a.dceNum
from dces a, exposures b, requests c, iracMap d
where a.reqkey = b.reqkey
and a.exposureNum = b.exposureNum
and b.reqkey = c.reqKey
and c.reqkey = d.reqKey
and c.reqTypeName = 'AOR'
and d.highDynamic = 't'
and a.initPlScriptId = 1
and a.reqKey = reqKey;

make = t
ensOfEns = f
minInputs = 2
comment = IRAC AOR HDR drk.
plScriptId(s) = 2

ruleId = 102
instrument = IRAC
sql = insert into EnsTempList
     (initPlScriptId, chanNum, fowlerNum, waitPeriod,
      dceNum)
     select unique
      a.initPlScriptId, a.chanNum, a.fowlerNum, a.waitPeriod,
a.dceNum
from dces a, exposures b, requests c
where a.reqkey = b.reqkey
and a.exposureNum = b.exposureNum
and b.reqkey = c.reqKey
and b.cycleNum is NULL
and c.reqTypeName = 'IER'
and a.initPlScriptId = 1
and a.reqKey = reqKey;

make = t
ensOfEns = f
minInputs = 2
comment = IRAC IER drk.
plScriptId(s) = 2
ruleId = 103
instrument = IRAC
sql = insert into EnsTempList
     (initPlScriptId, chanNum, fowlerNum, waitPeriod, dceNum)
     select unique
     a.initPlScriptId, a.chanNum, a.fowlerNum, a.waitPeriod, a.dceNum
     from dces a, exposures b, requests c
     where a.reqkey = b.reqkey
     and a.exposureNum = b.exposureNum
     and b.reqkey = c.reqKey
     and b.cycleNum is not NULL
     and c.reqTypeName = 'IER'
     and a.initPlScriptId = 1
     and a.reqKey = reqKey_;
make = t
ensOfEns = f
minInputs = 2
comment = IRAC IER drk.
plScriptId(s) = 2

ruleId = 110
instrument = IRAC
sql = insert into EnsTempList
     (initPlScriptId, chanNum)
     select unique
     initPlScriptId, chanNum
     from dces
     where initPlScriptId = 6
     and reqKey = reqKey_;
make = t
ensOfEns = f
minInputs = 2
comment = IRAC flt.
plScriptId(s) = 7, 8

ruleId = 115
instrument = IRAC
sql = insert into EnsTempList
     (initPlScriptId, chanNum, cycleNum)
     select unique
     a.initPlScriptId, a.chanNum, b.cycleNum
     from dces a, exposures b
     where a.reqkey = b.reqkey
     and a.exposureNum = b.exposureNum
     and b.cycleNum is not NULL
     and a.initPlScriptId = 17
     and a.reqKey = reqKey_;
make = t
ensOfEns = f
minInputs = 2
comment = IRAC sci hdr.
plScriptId(s) = 21

ruleId = 116
instrument = IRAC
sql = insert into EnsTempList
    (initPlScriptId, chanNum, cycleNum)
    select unique
    a.initPlScriptId, a.chanNum, b.cycleNum
    from dces a, exposures b
    where a.reqkey = b.reqkey
    and a.exposureNum = b.exposureNum
    and b.cycleNum is not NULL
    and a.initPlScriptId = 1
    and a.reqKey = reqKey_;
make = t
ensOfEns = f
minInputs = 2
comment = IRAC drk hdr.
plScriptId(s) = 21

ruleId = 117
instrument = IRAC
sql = insert into EnsTempList
    (initPlScriptId, chanNum, cycleNum)
    select unique
    a.initPlScriptId, a.chanNum, b.cycleNum
    from dces a, exposures b
    where a.reqkey = b.reqkey
    and a.exposureNum = b.exposureNum
    and b.cycleNum is not NULL
    and a.initPlScriptId = 6
    and a.reqKey = reqKey_;
make = t
ensOfEns = f
minInputs = 2
comment = IRAC flt hdr.
plScriptId(s) = 21

ruleId = 120
instrument = IRAC
sql = insert into EnsTempList
    (initPlScriptId, chanNum)
    select unique
    initPlScriptId, chanNum
    from dces
    where initPlScriptId = 17
    and reqKey = reqKey_;
make = t
ensOfEns = f
minInputs = 2
comment = IRAC lat.
plScriptId(s) = 23

ruleId = 121
instrument = IRAC
sql = insert into EnsTempList
    (initPlScriptId, chanNum)
    select unique
    initPlScriptId, chanNum
    from dces
where initPlScriptId = 1
and reqKey = reqKey_; 

make          = t
ensOfEns      = f
minInputs     = 2
comment       = IRAC lat.
plScriptId(s) = 23

ruleId       = 122
instrument   = IRAC
sql          = insert into EnsTempList
               (initPlScriptId, chanNum)
               select unique
               initPlScriptId, chanNum
               from dces
               where initPlScriptId = 6
               and reqKey = reqKey_; 

make          = t
ensOfEns      = f
minInputs     = 2
comment       = IRAC lat.
plScriptId(s) = 23

ruleId       = 125
instrument   = IRAC
sql          = insert into EnsTempList
               (initPlScriptId, clusterPosNum)
               select unique
               a.initPlScriptId, b.clusterPosNum
               from dces a, exposures b, targetFixedCluster c
               where a.reqkey = b.reqkey
               and a.exposureNum = b.exposureNum
               and a.reqkey = c.reqkey
               and a.primaryField = 1
               and c.arrayCoord = 'f'
               and a.initPlScriptId = 17
               and a.reqKey = reqKey_; 

make          = t
ensOfEns      = f
minInputs     = 2
comment       = IRAC reffr for science DCEs. For all DCEs with
plScriptid=17 (IRAC initial-processing pipeline for
science) in a given request, form DCE groups with the
same clusterPosNum (data for all channels combined).
plScriptId(s) = 25

ruleId       = 126
instrument   = IRAC
sql          = insert into EnsTempList
               (initPlScriptId)
               select unique
               initPlScriptId
               from dces
               where primaryField = 1
               and
               (reqkey not in
               (select reqkey
from targetfixedcluster)
or reqkey in
(select reqkey
from targetfixedcluster
where arraycoord = 't'))
and initPlScriptId = 17
and reqKey = reqKey_;
make = t
ensOfEns = f
minInputs = 2
comment = IRAC reffr for science DCEs.
plScriptId(s) = 25

ruleId = 130
instrument = IRAC
sql = insert into EnsTempList
(initPlScriptId, chanNum, clusterPosNum, arraycoord)
select unique
a.initPlScriptId, a.chanNum, b.clusterPosNum, 0
from dces a, exposures b, targetFixedCluster c
where a.reqkey = b.reqkey
and a.exposureNum = b.exposureNum
and a.reqkey = c.reqkey
and a.primaryField = 1
and c.arrayCoord = 'f'
and a.initPlScriptId = 17
and a.reqKey = reqKey_;
make = t
ensOfEns = f
minInputs = 2
comment = IRAC refin and mopex for science DCEs.
plScriptId(s) = 27, 34

ruleId = 131
instrument = IRAC
sql = insert into EnsTempList
(initPlScriptId, chanNum, arraycoord)
select unique
initPlScriptId, chanNum, 1
from dces
where primaryField = 1
and
(reqkey not in
(select reqkey
from targetfixedcluster)
or reqkey in
(select reqkey
from targetfixedcluster
where arraycoord = 't'))
and initPlScriptId = 17
and reqKey = reqKey_;
make = t
ensOfEns = f
minInputs = 2
comment = IRAC refin and mopex for science DCEs.
plScriptId(s) = 27, 34
ruleId = 135
instrument = IRAC
sql = insert into EnsTempList
    (initPlScriptId, clusterPosNum)
    select unique
    a.initPlScriptId, b.clusterPosNum
    from dces a, exposures b, targetFixedCluster c
    where a.reqkey = b.reqkey
    and a.exposureNum = b.exposureNum
    and a.reqkey = c.reqkey
    and a.primaryField = 1
    and c.arrayCoord = 'f'
    and a.initPlScriptId = 1
    and a.reqKey = reqKey_;
make = t
ensOfEns = f
minInputs = 2
comment = IRAC reffr for dark DCEs.
plScriptId(s) = 25

ruleId = 136
instrument = IRAC
sql = insert into EnsTempList
    (initPlScriptId)
    select unique
    initPlScriptId
    from dces
    where primaryField = 1
    and
    (reqkey not in
    (select reqkey
    from targetfixedcluster)
    or reqkey in
    (select reqkey
    from targetfixedcluster
    where arraycoord = 't'))
    and initPlScriptId = 1
    and reqKey = reqKey_;
make = t
ensOfEns = f
minInputs = 2
comment = IRAC reffr for dark DCEs.
plScriptId(s) = 25

ruleId = 140
instrument = IRAC
sql = insert into EnsTempList
    (initPlScriptId, chanNum, clusterPosNum, arraycoord)
    select unique
    a.initPlScriptId, a.chanNum, b.clusterPosNum, 'f'
    from dces a, exposures b, targetFixedCluster c
    where a.reqkey = b.reqkey
    and a.exposureNum = b.exposureNum
    and a.reqkey = c.reqkey
    and a.primaryField = 1
    and c.arrayCoord = 'f'
    and a.initPlScriptId = 1
    and a.reqKey = reqKey_;
and a.reqKey = reqKey_;  

dimension = t  
ensofEns = f  
minInputs = 2  
comment = IRAC refin and mopex for dark DCEs.  
plScriptId(s) = 27, 34  

ruleId = 141  
instrument = IRAC  
sql = insert into EnsTempList  
  (initPlScriptId, chanNum, arraycoord)  
select unique  
  initPlScriptId, chanNum, 1  
from dces  
where primaryField = 1  
and  
  (reqkey not in  
    (select reqkey  
     from targetfixedcluster)  
  or reqkey in  
    (select reqkey  
     from targetfixedcluster  
      where arraycoord = 't'))  
and initPlScriptId = 1  
and reqKey = reqKey_;  

make = t  
ensofEns = f  
mintput = 2  
comment = IRAC refin and mopex for dark DCEs.  
plScriptId(s) = 27, 34  

ruleId = 145  
instrument = IRAC  
sql = insert into EnsTempList  
  (initPlScriptId, clusterPosNum)  
select unique  
  a.initPlScriptId, b.clusterPosNum  
from dces a, exposures b, targetFixedCluster c  
where a.reqkey = b.reqkey  
and a.exposureNum = b.exposureNum  
and a.reqkey = c.reqkey  
and a.primaryField = 1  
and c.arrayCoord = 'f'  
and a.initPlScriptId = 6  
and a.reqKey = reqKey_;  

make = t  
ensofEns = f  
mintput = 2  
comment = IRAC reffr for flat DCEs.  
plScriptId(s) = 25  

ruleId = 146  
instrument = IRAC  
sql = insert into EnsTempList  
  (initPlScriptId)  
select unique  
  initPlScriptId
from dces
where primaryField = 1
and
(reqkey not in
(select reqkey
from targetfixedcluster)
or reqkey in
(select reqkey
from targetfixedcluster
where arraycoord = 't' ))
and initPlScriptId = 6
and reqKey = reqKey _;

make = t
ensOfEns = f
minInputs = 2
comment = IRAC reffr for flat DCEs.
plScriptId(s) = 25

ruleId = 150
instrument = IRAC
sql = insert into EnsTempList
(initPlScriptId, chanNum, clusterPosNum, arraycoord)
select unique
a.initPlScriptId, a.chanNum, b.clusterPosNum, 'f'
from dces a, exposures b, targetFixedCluster c
where a.regkey = b.regkey
and a.exposureNum = b.exposureNum
and a.regkey = c.regkey
and a.primaryField = 1
and c.arrayCoord = 'f'
and a.initPlScriptId = 6
and a.reqKey = reqKey _;

make = t
ensOfEns = f
minInputs = 2
comment = IRAC refin and mopex for flat DCEs.
plScriptId(s) = 27, 34

ruleId = 151
instrument = IRAC
sql = insert into EnsTempList
(initPlScriptId, chanNum, arraycoord)
select unique
initPlScriptId, chanNum, 1
from dces
where primaryField = 1
and
(reqkey not in
(select reqkey
from targetfixedcluster)
or reqkey in
(select reqkey
from targetfixedcluster
where arraycoord = 't' ))
and initPlScriptId = 6
and reqKey = reqKey _;

make = t
ensOfEns = f
minInputs = 2
comment = IRAC refin and mopex for flat DCEs.
plScriptId(s) = 27, 34
ruleId = 190
instrument = IRAC
sql = insert into EnsOfEnsTempList
     (ensPlScriptId, clusterPosNum)
     select unique
     b.ensPlScriptId, a.clusterPosNum
     from EnsTempList a, EnsTempList3 b, EnsTempListMore c
     where a.ruleId = b.ruleId
     and a.groupId = c.groupId
     and b.dceSetId = c.dceSetId
     and b.ensPlScriptId = 27
     and a.arraycoord = 0;
make = t
ensOfEns = t
minInputs = 2
comment = IRAC mrefi channels 1-4 (for multi-channel position
refinement). Associate together the “position
refinement” ensembles for all four channels in a given
request that have arraycoord='f' (in the
targetFixedCluster database table).
plScriptId(s) = 33
ruleId = 191
instrument = IRAC
sql = insert into EnsOfEnsTempList
     (ensPlScriptId)
     select unique
     b.ensPlScriptId
     from EnsTempList a, EnsTempList3 b, EnsTempListMore c
     where a.ruleId = b.ruleId
     and a.groupId = c.groupId
     and b.dceSetId = c.dceSetId
     and b.ensPlScriptId = 27
     and a.arraycoord = 1;
make = t
ensOfEns = t
minInputs = 2
comment = IRAC mrefi channels 1-4.
plScriptId(s) = 33
ruleId = 196
instrument = IRAC
sql = insert into EnsOfEnsTempList
     (ensPlScriptId, clusterPosNum)
     select unique
     b.ensPlScriptId, a.clusterPosNum
     from EnsTempList a, EnsTempList3 b, EnsTempListMore c
     where a.ruleId = b.ruleId
     and a.groupId = c.groupId
     and b.dceSetId = c.dceSetId
     and b.ensPlScriptId = 34
     and a.arraycoord = 0;
make          = t
ensOfEns      = t
minInputs     = 2
comment       = IRAC bmerg channels 1-4, split up by clusterPosNum.
plScriptId(s) = 32

ruleId        = 197
instrument    = IRAC
sql           = insert into EnsOfEnsTempList
                (ensPlScriptId)
                select unique
                b.ensPlScriptId
                from EnsTempList a, EnsTempList3 b, EnsTempListMore c
                where a.ruleId = b.ruleId
                and a.groupId = c.groupId
                and b.dceSetId = c.dceSetId
                and b.ensPlScriptId = 34
                and a.arraycoord = 1;

make          = t
ensOfEns      = t
minInputs     = 2
comment       = IRAC bmerg channels 1-4, independent of clusterPosNum.
plScriptId(s) = 32

ruleId        = 200
instrument    = MIPS
sql           = insert into EnsTempList
                (initPlScriptId)
                select unique
                a.initPlScriptId
                from dces a, exposures b
                where a.reqKey = b.reqKey
                and a.exposureNum = b.exposureNum
                and a.primaryField = 1
                and a.dceNum = 0
                and a.initPlScriptId = 1000
                and b.frameNum >= 2
                and b.readoutMode = 'SUR'
                and a.reqKey = reqKey_

make          = t
ensOfEns      = f
minInputs     = 3
comment       = MIPS-1 drk.
plScriptId(s) = 1001

ruleId        = 201
instrument    = MIPS
sql           = insert into EnsTempList
                (initPlScriptId, frameNum)
                select unique
                a.initPlScriptId, b.frameNum
                from dces a, exposures b
                where a.reqKey = b.reqKey
                and a.exposureNum = b.exposureNum
                and a.primaryField = 1
                and a.dceNum = 0
                and a.initPlScriptId = 1005
and b.frameNum >= 2
and b.readoutMode = 'RAW'
and a.reqKey = reqKey_;

make          = t
ensOfEns      = f
minInputs     = 3
comment       = MIPS-1 drk.
plScriptId(s) = 1006

ruleId        = 202
instrument    = MIPS
sql           = insert into EnsTempList
                (initPlScriptId)
                select unique
                a.initPlScriptId
                from dces a, exposures b
                where a.reqKey = b.reqKey
                and a.exposureNum = b.exposureNum
                and a.primaryField = 1
                and a.dceNum > 0
                and a.initPlScriptId = 1000
                and b.frameNum >= 2
                and b.readoutMode = 'SUR'
                and a.reqKey = reqKey_;

make          = t
ensOfEns      = f
minInputs     = 3
comment       = MIPS-1 drk.
plScriptId(s) = 1001

ruleId        = 203
instrument    = MIPS
sql           = insert into EnsTempList
                (initPlScriptId, frameNum)
                select unique
                a.initPlScriptId, b.frameNum
                from dces a, exposures b
                where a.reqKey = b.reqKey
                and a.exposureNum = b.exposureNum
                and a.primaryField = 1
                and a.dceNum > 0
                and a.initPlScriptId = 1005
                and b.frameNum >= 2
                and b.readoutMode = 'RAW'
                and a.reqKey = reqKey_;

make          = t
ensOfEns      = f
minInputs     = 3
comment       = MIPS-1 drk.
plScriptId(s) = 1006

ruleId        = 204
instrument    = MIPS
sql           = insert into EnsTempList
                (initPlScriptId)
                select unique
                a.initPlScriptId
from dces a, exposures b
where a.reqKey = b.reqKey
and a.exposureNum = b.exposureNum
and a.primaryField = 1
and a.dceNum > 0
and a.initPlScriptId = 1031
and b.frameNum >= 2
and b.readoutMode = 'SUR'
and a.reqKey = reqKey_; make          = t
ensOfEns      = f
minInputs     = 3
comment       = MIPS-1 pfl.
plScriptId(s) = 1032

ruleId        = 205
instrument    = MIPS
sql           = insert into EnsTempList
(initPlScriptId)
select unique
a.initPlScriptId
from dces a, exposures b
where a.reqKey = b.reqKey
and a.exposureNum = b.exposureNum
and a.primaryField = 1
and a.dceNum > 0
and a.initPlScriptId = 1033
and b.frameNum >= 2
and b.readoutMode = 'SUR'
and a.reqKey = reqKey_; make          = t
ensOfEns      = f
minInputs     = 3
comment       = MIPS-1 sfl.
plScriptId(s) = 1034

ruleId        = 207
instrument    = MIPS
sql           = insert into EnsTempList
(initPlScriptId)
select unique
a.initPlScriptId
from dces a, exposures b
where a.reqKey = b.reqKey
and a.exposureNum = b.exposureNum
and a.primaryField = 1
and a.dceNum > 0
and a.initPlScriptId = 1010
and b.frameNum = 5
and b.readoutMode = 'RAW'
and a.reqKey = reqKey_; make          = t
ensOfEns      = f
minInputs     = 3
comment       = MIPS-1 lin.
plScriptId(s) = 1011
ruleId = 208
instrument = MIPS
sql = insert into EnsTempList
   (initPlScriptId)
   select unique
   a.initPlScriptId
   from dces a, exposures b
   where a.reqKey = b.reqKey
   and a.exposureNum = b.exposureNum
   and a.primaryField = 1
   and a.dceNum > 0
   and a.initPlScriptId = 1015
   and b.frameNum >= 2
   and b.readoutMode = 'SUR'
   and a.reqKey = reqKey_

make = t
ensOfEns = f
minInputs = 3
comment = MIPS-1 flt.
plScriptId(s) = 1016

ruleId = 210
instrument = MIPS
sql = insert into EnsTempList
   (exposureNum)
   select unique
   exposureNum
   from dces
   where primaryField < 3
   and
   (initPlScriptId = 1020
   or initPlScriptId = 1021)
   and reqKey = reqKey_

make = t
ensOfEns = f
minInputs = 2
comment = MIPS-1 lat.
plScriptId(s) = 1035

ruleId = 215
instrument = MIPS
sql = insert into EnsTempList
   (clusterPosNum, primaryField)
   select unique
   nvl(b.clusterPosNum, -1),
   a.primaryField
   from dces a, exposures b
   where a.reqKey = b.reqKey
   and a.exposureNum = b.exposureNum
   and
   (a.initPlScriptId = 1020
   or a.initPlScriptId = 1605
   or a.initPlScriptId = 1606
   or a.initPlScriptId = 1607
   or a.initPlScriptId = 1608
   or a.initPlScriptId = 1609)
   and a.reqKey = reqKey_
make = t
ensOfEns = f
minInputs = 2
comment = MIPS reffr for scn exposureTypes for chanNum<=2.
plScriptId(s) = 1040

ruleId = 220
instrument = MIPS
sql = insert into EnsTempList
    (clusterPosNum, primaryField)
    select unique
    nvl(b.clusterPosNum, -1),
    a.primaryField
    from dces a, exposures b
    where a.reqKey = b.reqKey
    and a.exposureNum = b.exposureNum
    and
    (a.initPlScriptId = 1021
    or a.initPlScriptId = 1600
    or a.initPlScriptId = 1601
    or a.initPlScriptId = 1602
    or a.initPlScriptId = 1603
    or a.initPlScriptId = 1604)
    and a.reqKey = reqKey_;

make = t
ensOfEns = f
minInputs = 2
comment = MIPS reffr for pht exposureTypes for chanNum<=2.
plScriptId(s) = 1040

ruleId = 225
instrument = MIPS
sql = insert into EnsTempList
    (initPlScriptId, clusterPosNum, primaryField)
    select unique
    a.initPlScriptId, nvl(b.clusterPosNum, -1),
    a.primaryField
    from dces a, exposures b
    where a.reqKey = b.reqKey
    and a.exposureNum = b.exposureNum
    and
    a.initPlScriptId = 1020
    and a.reqKey = reqKey_;

make = t
ensOfEns = f
minInputs = 2
comment = MIPS-1 refin for scn exposureTypes.
plScriptId(s) = 1042

ruleId = 226
instrument = MIPS
sql = insert into EnsTempList
    (initPlScriptId, clusterPosNum, primaryField)
    select unique
    a.initPlScriptId, nvl(b.clusterPosNum, -1),
    a.primaryField
    from dces a, exposures b
    where a.reqKey = b.reqKey
and a.exposureNum = b.exposureNum
and a.initPlScriptId = 1021
and a.reqKey = reqKey_;
make = t
ensOfEns = f
minInputs = 2
comment = MIPS-1 refin for pht exposureTypes.
plScriptId(s) = 1042
ruleId = 236
instrument = MIPS
sql = insert into EnsTempList
  (initPlScriptId, clusterPosNum, primaryField, chanNum)
  select unique
    a.initPlScriptId, nvl(b.clusterPosNum, -1),
    a.primaryField, a.chanNum
  from dces a, exposures b
  where a.reqKey = b.reqKey
  and a.exposureNum = b.exposureNum
  and a.initPlScriptId = 1020
  and a.dceNum > 0
  and a.reqKey = reqKey_;
make = t
ensOfEns = f
minInputs = 2
comment = MIPS-1 mopex for scn exposureTypes.
plScriptId(s) = 1048
ruleId = 237
instrument = MIPS
sql = insert into EnsTempList
  (initPlScriptId, clusterPosNum, primaryField, chanNum)
  select unique
    a.initPlScriptId, nvl(b.clusterPosNum, -1),
    a.primaryField, a.chanNum
  from dces a, exposures b
  where a.reqKey = b.reqKey
  and a.exposureNum = b.exposureNum
  and a.initPlScriptId = 1021
  and a.dceNum > 0
  and a.reqKey = reqKey_;
make = t
ensOfEns = f
minInputs = 2
comment = MIPS-1 mopex for pht exposureTypes.
plScriptId(s) = 1048
ruleId = 300
instrument = MIPS
sql = insert into EnsTempList
  (initPlScriptId, clusterPosNum, frameNum, primaryField)
  select unique
    a.initPlScriptId, nvl(b.clusterPosNum, -1),
    b.frameNum, a.primaryField
  from dces a, exposures b
  where a.reqKey = b.reqKey
  and a.exposureNum = b.exposureNum
and
(a.initPlScriptId >= 1500
and a.initPlScriptId <= 1504)
and a.reqKey = reqKey_;
and a.exposureNum = b.exposureNum
and
(a.initPlScriptId >= 1700
and a.initPlScriptId <= 1704)
and a.reqKey = reqKey_;

make = t
ensOfEns = f
minInputs = 2
comment = MIPS-3 pht for primaryField=1.
plScriptId(s) = -100

ruleId = 343
instrument = MIPS
sql = insert into EnsTempList
  (initPlScriptId, exposureNum, frameNum)
select unique
  a.initPlScriptId, b.exposureNum, b.frameNum
from dces a, exposures b
where a.reqKey = b.reqKey
and a.exposureNum = b.exposureNum
and
(a.initPlScriptId >= 1705
and a.initPlScriptId <= 1709)
and a.reqKey = reqKey_;

make = t
ensOfEns = f
minInputs = 2
comment = MIPS-3 various.
plScriptId(s) = -100

ruleId = 344
instrument = MIPS
sql = insert into EnsTempList
  (initPlScriptId, frameNum)
select unique
  a.initPlScriptId, b.frameNum
from dces a, exposures b
where a.reqKey = b.reqKey
and a.exposureNum = b.exposureNum
and
(a.initPlScriptId >= 1710
and a.initPlScriptId <= 1799)
and a.reqKey = reqKey_;

make = t
ensOfEns = f
minInputs = 2
comment = MIPS-3 various.
plScriptId(s) = -100

ruleId = 345
instrument = MIPS
sql = insert into EnsTempList
  (clusterPosNum, chanNum)
select unique
  nvl(b.clusterPosNum, -1),
  a.chanNum
from dces a, exposures b
where a.reqKey = b.reqKey
and a.exposureNum = b.exposureNum
and
(a.initPlScriptId >= 1500
and a.initPlScriptId <= 1509)
and a.primaryField = 1
and a.reqKey = reqKey_;
b.ensPlScriptId, a.clusterPosNum
from EnsTempList a, EnsTempList3 b, EnsTempListMore c
where a.ruleId = b.ruleId
and a.groupId = c.groupId
and b.dceSetId = c.dceSetId
and b.ensPlScriptId = 1048
and a.chanNum <= 2
and a.chanNum >= 1;

make          = t
ensOfEns      = t
minInputs     = 2
comment       = MIPS bmerg channels 1-2.
plScriptId(s) = 1047
Appendix C: Test-Mode Sample Output from createEnsembles.pl

As a fine point of interest, this case is for a MIPS campaign request that includes the combining of “right” and “left” MIPS:Ge dark ensembles into an “ensemble of ensembles”.

% createEnsembles.pl –r 3861504 –o req6758656.out –t
% cat req6758656.out
createEnsembles.pl, Version 1.0
By Russ Laher (laher@ipac.caltech.edu)
Copyright (C) 2004
Spitzer-Space-Telescope Science Center (SSC),
California Institute of Technology, Pasadena, CA

For a given request, creates the ensembles, dceSets, and ensembleSets records specified by the information in the database ensRules and ensPlScripts tables. Optionally lists the ensemble-creation rules, either for a given instrument or for all instruments. A test mode is available to verify the correctness of the ensemble-creation rules without actually creating the ensembles, dceSets, and ensembleSets records.

Database, role = sodb_dnl15, dnlDevRole
Date/time = Fri May 7 07:27:46 2004
Request = 6758656
Pre-loading data for dceSets and ensembles records...

(Test mode: ensId < 0 means that the ensembles were not actually created.)

------------------------------------ruleId = 200

DCE Sets:
ruleId  initPlScriptId  dceSetId  expectedInputs  representativeDceId
200     1000     15195                  5              2185687

Ensembles:
ruleId  ensId  ensPlScriptId  dceId  dceSetId  expectedInputs
200     -1      1001     2185687       15195           5

------------------------------------ruleId = 202

DCE Sets:
ruleId  initPlScriptId  dceSetId  expectedInputs  representativeDceId
202     1000     15196                  45              2185688

Ensembles:
ruleId  ensId  ensPlScriptId  dceId  dceSetId  expectedInputs
202     -2      1001     2185688       15196           45
---ruleId = 321---

DCE Sets:

<table>
<thead>
<tr>
<th>ruleId</th>
<th>initPlScriptId</th>
<th>frameNum</th>
<th>dceSetId</th>
<th>expectedInputs</th>
<th>representativeDceId</th>
</tr>
</thead>
<tbody>
<tr>
<td>321</td>
<td>1540</td>
<td>2</td>
<td>15197</td>
<td>101</td>
<td>2185838</td>
</tr>
<tr>
<td>321</td>
<td>1545</td>
<td>2</td>
<td>15198</td>
<td>101</td>
<td>2186141</td>
</tr>
</tbody>
</table>

Ensembles:

<table>
<thead>
<tr>
<th>ruleId</th>
<th>ensId</th>
<th>ensPlScriptId</th>
<th>dceId</th>
<th>dceSetId</th>
<th>expectedInputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>321</td>
<td>-3</td>
<td>1640</td>
<td>2185838</td>
<td>15197</td>
<td>101</td>
</tr>
<tr>
<td>321</td>
<td>-4</td>
<td>1645</td>
<td>2186141</td>
<td>15198</td>
<td>101</td>
</tr>
</tbody>
</table>

---ruleId = 344---

DCE Sets:

<table>
<thead>
<tr>
<th>ruleId</th>
<th>initPlScriptId</th>
<th>frameNum</th>
<th>dceSetId</th>
<th>expectedInputs</th>
<th>representativeDceId</th>
</tr>
</thead>
<tbody>
<tr>
<td>344</td>
<td>1730</td>
<td>2</td>
<td>15199</td>
<td>101</td>
<td>2186445</td>
</tr>
<tr>
<td>344</td>
<td>1735</td>
<td>2</td>
<td>15200</td>
<td>101</td>
<td>2185839</td>
</tr>
</tbody>
</table>

Ensembles:

<table>
<thead>
<tr>
<th>ruleId</th>
<th>ensId</th>
<th>ensPlScriptId</th>
<th>dceId</th>
<th>dceSetId</th>
<th>expectedInputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>344</td>
<td>-5</td>
<td>1830</td>
<td>2186445</td>
<td>15199</td>
<td>101</td>
</tr>
<tr>
<td>344</td>
<td>-6</td>
<td>1835</td>
<td>2185839</td>
<td>15200</td>
<td>101</td>
</tr>
</tbody>
</table>

Total number of ensembles created = 6

Pre-loading data for ensembleSets and associated ensembles records...

(Test mode: ensId < 0 means that the ensembles were not actually created.)

---ruleId = 350---

Ensemble Sets:

<table>
<thead>
<tr>
<th>ruleId</th>
<th>outEnsId</th>
<th>inEnsId</th>
</tr>
</thead>
<tbody>
<tr>
<td>350</td>
<td>-1000000</td>
<td>-4</td>
</tr>
<tr>
<td>350</td>
<td>-1000000</td>
<td>-3</td>
</tr>
</tbody>
</table>

Ensembles:

<table>
<thead>
<tr>
<th>ruleId</th>
<th>ensId</th>
<th>ensPlScriptId</th>
<th>expectedInputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>350</td>
<td>-1000000</td>
<td>1645</td>
<td>202</td>
</tr>
</tbody>
</table>

Total number of ensemble sets created = 1
Appendix D: Normal-Mode Sample Output from createEnsembles.pl

This case is for an IRAC campaign request, and it includes the combining of ensembles across the four IRAC channels into “ensembles of ensembles” for multi-band pointing refinement and band-merging.

% createEnsembles.pl -r 3861504 -o req3861504.out
% cat req3861504.out

createEnsembles.pl, Version 1.0
By Russ Laher (laher@ipac.caltech.edu)
Copyright (C) 2004
Spitzer-Space-Telescope Science Center (SSC),
California Institute of Technology, Pasadena, CA

For a given request, creates the ensembles, dceSets, and ensembleSets records specified by the information in the database ensRules and ensPlScripts tables. Optionally lists the ensemble-creation rules, either for a given instrument or for all instruments. A test mode is available to verify the correctness of the ensemble-creation rules without actually creating the ensembles, dceSets, and ensembleSets records.

Database, role = sodb_dn15, dn1DevRole
Date/time = Tue May  4 07:08:25 2004
Request = 3861504

Creating dceSets and ensembles records...

------------------------------------ruleId = 115

DCE Sets:

ruleId initPlScriptId chanNum cycleNum dceSetId expectedInputs representativeDceId
115 17 1 1 15157 2 5193491
115 17 2 1 15158 2 5193490
115 17 3 1 15159 2 5193489
115 17 4 1 15160 2 5193488

Ensembles:

ruleId ensId ensPlScriptId dceId dceSetId expectedInputs
115 17719 21 5193491 15157 2
115 17720 21 5193490 15158 2
115 17721 21 5193489 15159 2
115 17722 21 5193488 15160 2

------------------------------------ruleId = 120

DCE Sets:

ruleId initPlScriptId chanNum dceSetId expectedInputs representativeDceId
120 17 1 15161 316 5193491
120 17 2 15162 316 5193490
120 17 3 15163 316 5193489
120 17 4 15164 316 5193488

Ensembles:

ruleId ensId ensPlScriptId dceId dceSetId expectedInputs
120 17723 23 5193491 15161 316
120 17724 23 5193490 15162 316
120 17725 23 5193489 15163 316
120 17726 23 5193488 15164 316
DCE Sets:

<table>
<thead>
<tr>
<th>ruleId</th>
<th>initPlScriptId</th>
<th>dceSetId</th>
<th>expectedInputs</th>
<th>representativeDceId</th>
</tr>
</thead>
<tbody>
<tr>
<td>126</td>
<td>17</td>
<td>15165</td>
<td>1264</td>
<td>5193488</td>
</tr>
</tbody>
</table>

Ensembles:

<table>
<thead>
<tr>
<th>ruleId</th>
<th>ensId</th>
<th>ensPlScriptId</th>
<th>dceId</th>
<th>dceSetId</th>
<th>expectedInputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>126</td>
<td>17727</td>
<td>25</td>
<td>5193488</td>
<td>15165</td>
<td>1264</td>
</tr>
</tbody>
</table>

DCE Sets:

<table>
<thead>
<tr>
<th>ruleId</th>
<th>initPlScriptId</th>
<th>chanNum</th>
<th>arraycoord</th>
<th>dceSetId</th>
<th>expectedInputs</th>
<th>representativeDceId</th>
</tr>
</thead>
<tbody>
<tr>
<td>131</td>
<td>17</td>
<td>1</td>
<td>1</td>
<td>15166</td>
<td>316</td>
<td>5193491</td>
</tr>
<tr>
<td>131</td>
<td>17</td>
<td>2</td>
<td>1</td>
<td>15167</td>
<td>316</td>
<td>5193490</td>
</tr>
<tr>
<td>131</td>
<td>17</td>
<td>3</td>
<td>1</td>
<td>15168</td>
<td>316</td>
<td>5193489</td>
</tr>
<tr>
<td>131</td>
<td>17</td>
<td>4</td>
<td>1</td>
<td>15169</td>
<td>316</td>
<td>5193488</td>
</tr>
</tbody>
</table>

Ensembles:

<table>
<thead>
<tr>
<th>ruleId</th>
<th>ensId</th>
<th>ensPlScriptId</th>
<th>dceId</th>
<th>dceSetId</th>
<th>expectedInputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>131</td>
<td>17728</td>
<td>27</td>
<td>5193491</td>
<td>15166</td>
<td>316</td>
</tr>
<tr>
<td>131</td>
<td>17729</td>
<td>34</td>
<td>5193491</td>
<td>15166</td>
<td>316</td>
</tr>
<tr>
<td>131</td>
<td>17730</td>
<td>27</td>
<td>5193490</td>
<td>15167</td>
<td>316</td>
</tr>
<tr>
<td>131</td>
<td>17731</td>
<td>34</td>
<td>5193490</td>
<td>15167</td>
<td>316</td>
</tr>
<tr>
<td>131</td>
<td>17732</td>
<td>27</td>
<td>5193489</td>
<td>15168</td>
<td>316</td>
</tr>
<tr>
<td>131</td>
<td>17733</td>
<td>34</td>
<td>5193489</td>
<td>15168</td>
<td>316</td>
</tr>
<tr>
<td>131</td>
<td>17734</td>
<td>27</td>
<td>5193488</td>
<td>15169</td>
<td>316</td>
</tr>
<tr>
<td>131</td>
<td>17735</td>
<td>34</td>
<td>5193488</td>
<td>15169</td>
<td>316</td>
</tr>
</tbody>
</table>

Total number of ensembles created = 17

Creating ensembleSets and associated ensembles records...

DCE Sets:

<table>
<thead>
<tr>
<th>ruleId</th>
<th>initPlScriptId</th>
<th>chanNum</th>
<th>arraycoord</th>
<th>dceSetId</th>
<th>expectedInputs</th>
<th>representativeDceId</th>
</tr>
</thead>
<tbody>
<tr>
<td>131</td>
<td>17</td>
<td>1</td>
<td>1</td>
<td>15166</td>
<td>316</td>
<td>5193491</td>
</tr>
<tr>
<td>131</td>
<td>17</td>
<td>2</td>
<td>1</td>
<td>15167</td>
<td>316</td>
<td>5193490</td>
</tr>
<tr>
<td>131</td>
<td>17</td>
<td>3</td>
<td>1</td>
<td>15168</td>
<td>316</td>
<td>5193489</td>
</tr>
<tr>
<td>131</td>
<td>17</td>
<td>4</td>
<td>1</td>
<td>15169</td>
<td>316</td>
<td>5193488</td>
</tr>
</tbody>
</table>

Ensembles:

<table>
<thead>
<tr>
<th>ruleId</th>
<th>ensId</th>
<th>ensPlScriptId</th>
<th>dceId</th>
<th>dceSetId</th>
<th>expectedInputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>131</td>
<td>17728</td>
<td>27</td>
<td>5193491</td>
<td>15166</td>
<td>316</td>
</tr>
<tr>
<td>131</td>
<td>17729</td>
<td>34</td>
<td>5193491</td>
<td>15166</td>
<td>316</td>
</tr>
<tr>
<td>131</td>
<td>17730</td>
<td>27</td>
<td>5193490</td>
<td>15167</td>
<td>316</td>
</tr>
<tr>
<td>131</td>
<td>17731</td>
<td>34</td>
<td>5193490</td>
<td>15167</td>
<td>316</td>
</tr>
<tr>
<td>131</td>
<td>17732</td>
<td>27</td>
<td>5193489</td>
<td>15168</td>
<td>316</td>
</tr>
<tr>
<td>131</td>
<td>17733</td>
<td>34</td>
<td>5193489</td>
<td>15168</td>
<td>316</td>
</tr>
<tr>
<td>131</td>
<td>17734</td>
<td>27</td>
<td>5193488</td>
<td>15169</td>
<td>316</td>
</tr>
<tr>
<td>131</td>
<td>17735</td>
<td>34</td>
<td>5193488</td>
<td>15169</td>
<td>316</td>
</tr>
</tbody>
</table>

Total number of ensemble sets created = 2