Association Loader Design

Author: Dave Miers

Created: March 29, 2005

Last Modified: March 29, 2005 13:27

1 Purpose of Document

The purpose of this document is to define the design for the association loader.

2 Introduction

There are times when accession IDs (sequence IDs, clone ID, etc) need to be associated with MGI objects (clones, markers, etc). The *ACC_Accession* table in the MGD database is used to maintain these associations. The association loader uses records in the *MGI_Association* table in the RADAR database to create new associations in the *ACC_Accession* table. There are two different ways to use the association loader.

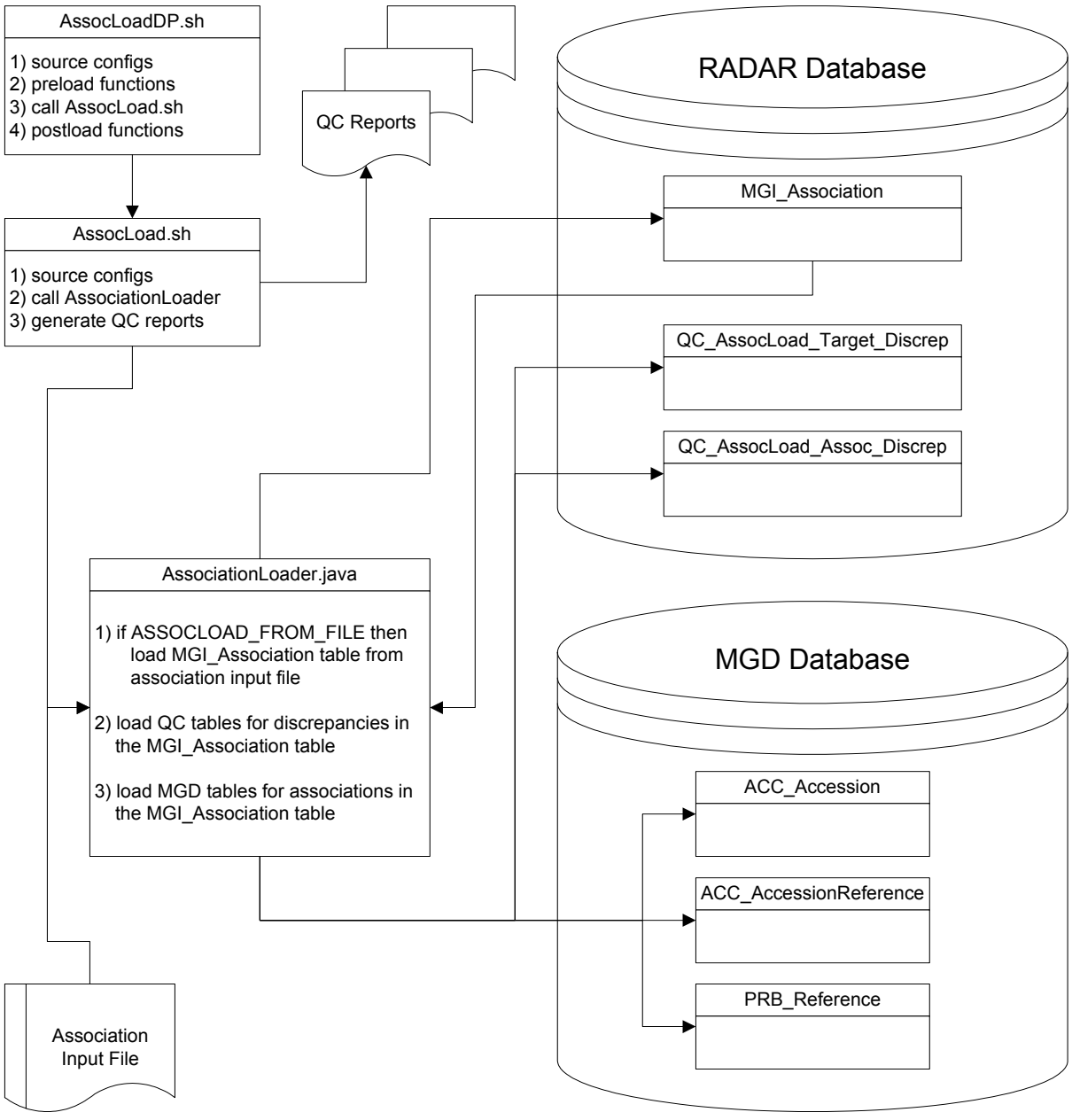
The first method is when the association loader act only as an associator. It expects that the *MGI_Association* table has been loaded by some other application (data provider loader) and it only needs to use the data in this table to create associations.

The second method is when the association loader acts as a data provider loader and an associator. The only difference with this method is that it has a preliminary step where it loads the *MGI_Association* table using an association input file. Then it proceeds with its normal processing to create the associations (as described above in the first method). The association loader design is specified in this document using the following sections:

- Responsibilities - Provide a high level list of the tasks that need to be accomplished.
- Invocation - Provides a description of the two different ways that the association loader can be invoked.
- Inputs - Provide a description of the inputs expected by the association loader.
- Outputs - Provide a description of the outputs that need to be generated.
- Java Objects - Provide a description of important Java objects that are reference in the processing section.
- Processing - Provide a description of the sequence of steps that must be taken to create the new associations
- Exceptions - Provide a list of the conditions that will cause the application to throw an exception.

The CVS product for the association loader is **assocload**.

The following diagram shows the interaction of the main components of the association loader.



3 Related Documents

This document assumes that the reader has read and understands the following documents:

- DLA Standards
- Java Frameworks Documentation

4 Definitions

The following definitions describe some of the terminology used in this document:

- **Association** - The relationship that an accession ID has with a MGI object that is represented by a record in the *ACC_Accession* table.
- **Data Provider Loader** - A load that is responsible for loading the *MGI_Association* table with data for the association loader to process.
- **Target Object** - The MGI object that is intended to have new associations made to it.

See the DLA Standards document for additional definitions of terminology.

5 Assumptions

The following assumptions are made for this document:

- The Java Frameworks is used to implement the load.

6 Responsibilities

The following responsibilities are handled by the association loader:

1. Load the *MGI_Association* table from an association input file (**optional**). This is only done when the association loader is used as a data provider loader.
2. Delete any associations made by the last invocation of the association loaded by a given job stream (if the delete/reload option is enabled).
3. Retrieve records from the *MGI_Association* table in the RADAR database and generate association records to load into the *ACC_Accession* table the MGD database.
4. For each record created for the *ACC_Accession* table, create a corresponding record in the *ACC_AccessionReference* table.
5. For clone associations ONLY, create a record for the *PRB_Reference* table for each clone that has one or more new associations made to it.
6. Provide all required reports defined in the Output section of this document.
7. Provide all the necessary log files as defined by the DLA standards.

7 Invocation

There are two different ways to use the association loader. It can be used just as an associator or it can be used as a data provider loader and associator together.

7.1 Associator Invocation

When the association loader is only being used as an associator, it is invoked by calling the shell script wrapper *AssocLoad.sh*. This shell script will invoke the Java application to use records in the *MGI_Association* table to create associations in the *ACC_Accession* table. The shell script is called as follows:

AssocLoad.sh ConfigFile JobKey [SystemProps]

where

Config File is the path name of the configuration file for the data provider loader.

JobKey is the value of *MGI_Association._JobStream_key* that identifies the records in the table that should be processed by this job stream.

SystemProps is a list of optional system properties to be passed to the Java application (for example -DSYSPROP1=abc -DSYSPROP2=123).

7.2 Data Provider Loader and Associator Invocation

When the association loader is being used as a data provider loader and an associator, it is invoked by calling the shell script wrapper *AssocLoadDP.sh*. This shell script will eventually call the *AssocLoad.sh* wrapper. The shell script is called as follows:

AssocLoadDP.sh ConfigFile JobKey [SystemProps]

where

Config File is the path name of the configuration file that contains the data provider settings required by the association loader.

JobKey is the value to be used for *MGI_Association._JobStream_key* when associations from the data provider input file are loaded into the *MGI_Association* table.

SystemProps is a list of optional system properties to be passed to the Java application (for example -DSYSPROP1=abc -DSYSPROP2=123).

8 Inputs

8.1 Configuration Variables

Configuration variables that are defined in the configuration file(s) are used by the association loader during execution. The following table lists the configuration variables that are specific to the association loader and must be defined.

Table 1: Configuration Variables

Name	Description	Required
ASSOCLOAD_FROM_FILE	Indicates whether a data provider input file needs to be loaded into the *MGI_Association* table (true) or if a data provider has already loaded this table (false). The default is false.	No
INFILE_NAME	The path name of the association input file to be loaded by the association loader when the ASSOCLOAD_FROM_FILE variable is true.	No
ASSOCLOAD_DELETE_RELOAD	Indicates whether associations from the prior load should be deleted first (true/false). The default is false.	No
ASSOCLOAD_TARGET_MGI_TYPE	The type of object that the target accession ID should be associated with in MGI. It should be a name from the *ACC_MGIType* table.	Yes
ASSOCLOAD_SINGLE_OBJECT_DB	A comma-separated list of logical DB names that should be associated with only one MGI object.	Yes
ASSOCLOAD_MULTIPLE_OBJECT_DB	A comma-separated list of logical DB names that can be associated with one or more MGI objects.	Yes

System properties are passed to the association loader when it is invoked and may override configuration variables set in a configuration file. The following table lists the system properties used by the association loader:

Table 2: System Properties

Name	Description	Required
CONFIG	A comma-separated list of configuration file path names that the association loader should use to get its configuration values. The files are listed in reverse order of precedence.	Yes
JOBKEY	The job key for the current job stream. This identifies the records in the *MGI_Association* table to be processed.	Yes

8.2 Input File

When the association loader is being used as a data provider loader, it will attempt to load the *MGI_Association* table using the input file defined by the environment variable INFILE_NAME. This file always has the same format, regardless of the type of associations that are being defined in it. There is one header record that defines the logical DBs for each column of data in the detail records. There is one detail record for each MGI object that is having accession IDs associated to it. The format of the file is as follows:

Target_DB <TAB> External_DB_1 ..... <TAB> External_DB_n

Target_ID <TAB> External_ID_List_1 ..... <TAB> External_ID_List_n

Target_ID <TAB> External_ID_List_1 ..... <TAB> External_ID_List_n

where

Target_DB is the logical DB for target accession ID in the *Target_ID* field of each detail record.

External_DB_1 is the logical DB for the accession ID(s) in the *External_ID_List_1* field of each detail record.

External_DB_n is the logical DB for the accession ID(s) in the *External_ID_List_n* field of each detail record. This is an optional field.

Target_ID is the target accession ID that is expected to be associated with one (and only one) target MGI object of the expected type (defined by ASSOCLOAD_TARGET_MGI_TYPE).

External_ID_List_1 is a comma-separated list of accession IDs for *External_DB_1* that are to be associated with the target MGI object.

External_ID_List_n is a comma-separated list of accession IDs for *External_DB_n* that are to be associated with the target MGI object.

8.3 MGI Association Table

When the *MGI_Association* table has been loaded by another data provider loader, it is used as input to the association loader. Each set of records in this table with the same *_Record_key* define one or more associations to be made to an object in MGI.

Table 3: MGI_Association

Column Name	Type	Description
_Assoc_key	int	This column contains the primary key for the table.
_Record_key	int	This column contains the key that identifies all the records for this load that originated from the same input record and must be processed together.
accID	varchar(30)	This column contains an accession ID.
logicalDB	varchar(80)	This column contains the logical DB name for the accession ID.
target	bit	This column indicates whether the accession ID for this record represents the target MGI object (1) or an accession ID to be associated with the target (0).
_JobStream_key	int	This column contains the key for the current job stream.
creation_date	datetime	This column contains the date/time when the record was loaded.

9 Outputs

9.1 BCP Files

The association loader produces a set of bcp files that are used to load tables in the RADAR and MGD databases. Most of the information written to the bcp files comes from a MGI association object (MAO) used during processing. See “MGI Association Object (MAO)” on page 17. for more information about this object. The following tables describe what values are used to populate each field in the bcp files.

9.1.1 MGI_Association.bcp

This file is only used when the association loader takes on the responsibility of a data provider loader and needs to load the *MGI_Association* table from an association input file.

Table 4: MGI_Association.bcp

Field Name	Value For A New Association Record
_Assoc_key	Populate with the next sequential primary key.
_Record_key	Populate with the same sequential key for each accession ID that came from the same input record.
accID	Populate with an accession ID from an input record.
logicalDB	Populate with the logical DB from an input record for the accession ID. The logical DB comes from the field in the header record of the input file that corresponds to the field that the accession ID came from.
target	Populate with a 1 if this accession ID is the target accession ID, otherwise 0.
_JobStream_key	Automatically populated by the Java Frameworks.
creation_date	Automatically populated by the Java Frameworks.

9.1.2 ACC_Accession.bcp

This file is used to load new associations into the *ACC_Accession* table.

Table 5: ACC_Accession.bcp

Field Name	Value For A New Association Record
_Accession_key	Populate with the next sequential primary key. This key is also used for the corresponding record in the *ACC_AccessionReference* table.
accID	Populate with an accession ID from the MGI association object that is being associated with the target MGI object.
prefixPart	Populate with the leading part of the accession ID up to and including the last non-numeric character.
numericPart	Populate with the trailing numeric characters of the accession ID after the *prefixPart*.

Table 5: ACC_Accession.bcp

Field Name	Value For A New Association Record
_LogicalDB_key	Populate with the logical DB key for the accession ID in the MGI association object.
_Object_key	Populate with the primary key of the MGI object that the target accession ID in the MGI association object is associated with.
_MGIType_key	Populate with the primary key for the target MGI type identified by the configuration variable ASSOCLOAD_TARGET_MGI_TYPE.
private	Populate with a 0.
preferred	Populate with a 1.
_CreatedBy_key	Automatically populated by the Java Frameworks.
_ModifiedBy_key	Automatically populated by the Java Frameworks.
creation_date	Automatically populated by the Java Frameworks.
modification_date	Automatically populated by the Java Frameworks.

9.1.3 ACC_AccessionReference.bcp

This file is used to load the corresponding reference records into the *ACC_AccessionReference* table for each association that is made in the *ACC_Accession* table.

Table 6: ACC_AccessionReference.bcp

Field Name	Value For A New Association Reference Record
_Accession_key	Populate with the *_Accession_key* from the corresponding record in the *ACC_Accession* table.
_Refs_key	Populate with the primary key for the reference (J-Number) that represents the current job stream.
_CreatedBy_key	Automatically populated by the Java Frameworks.
_ModifiedBy_key	Automatically populated by the Java Frameworks.
creation_date	Automatically populated by the Java Frameworks.
modification_date	Automatically populated by the Java Frameworks.

9.1.4 PRB_Reference.bcp

If the load is making associations to clone objects, this file is used to load reference records into the *PRB_Reference* table for each clone that has a new association made to it.

Table 7: PRB_Reference.bcp

Field Name	Value For A New Clone Association Reference Record
_Reference_key	Populate with the next sequential primary key.
_Probe_key	Populate with the primary key for the clone that is having a new association made to it.
_Refs_key	Populate with the primary key for the reference (J-Number) that represents the current job stream.
hasRmap	Populate with a 0.
hasSequence	Populate with a 0.
_CreatedBy_key	Automatically populated by the Java Frameworks.
_ModifiedBy_key	Automatically populated by the Java Frameworks.
creation_date	Automatically populated by the Java Frameworks.
modification_date	Automatically populated by the Java Frameworks.

9.1.5 QC_AssocLoad_Target_Discrep.bcp

This file is used to load QC report data into the *QC_AssocLoad_Target_Discrep* table. See “Target Discrepancy Report” on page 12. for more information about this report.

Table 8: QC_AssocLoad_Target_Discrep.bcp

Field Name	Value
_QCReport_key	Populate with the next sequential primary key.
accID	Populate with the target accession ID from the MGI association object.
_LogicalDB_key	Populate with the target logical DB key in the MGI association object.
_MGIType_key	Populate with the MGI type key of the MGI object that the target accession ID in the MGI association object is associated with. This field will be null if no association exists.

Table 8: QC_AssocLoad_Target_Discrep.bcp

Field Name	Value
_Object_key	Populate with the primary key of the MGI object that the target accession ID in the MGI association object is associated with. This field will be null if no association exists.
expectedType	Populate with the target MGI type identified by the configuration variable ASSOCLOAD_TARGET_MGI_TYPE.
message	Populate with the discrepancy message. See "Target Discrepancy Report Messages" on page 13.
_JobStream_key	Automatically populated by the Java Frameworks.
creation_date	Automatically populated by the Java Frameworks.

9.1.6 QC_AssocLoad_Assoc_Discrep.bcp

This file is used to load QC report data into the *QC_AssocLoad_Assoc_Discrep* table. See "Associate Discrepancy Report" on page 14. for more information about this report.

Table 9: QC_AssocLoad_Assoc_Discrep.bcp

Field Name	Value
_QCReport_key	Populate with the next sequential primary key.
tgtAccID	Populate with the target accession ID from the MGI association object.
_TgtLogicalDB_key	Populate with the target logical DB key in the MGI association object.
_TgtMGIType_key	Populate with the MGI type key of the MGI object that the target accession ID in the MGI association object is associated with.
_TgtObject_key	Populate with the primary key of the MGI object that the target accession ID in the MGI association object is associated with.
accID	Populate with the accession ID from the MGI association object that has a discrepancy with regard to the target MGI object.
_LogicalDB_key	Populate with the logical DB key for the accession ID in the MGI association object that has a discrepancy with regard to the target MGI object.

Table 9: QC_AssocLoad_Assoc_Discrep.bcp

Field Name	Value
_MGIType_key	Populate with the MGI type key of the MGI object that the accession ID is associated with that has a discrepancy with regard to the target MGI object.
_Object_key	Populate with the primary key of the MGI object that the accession ID is associated with that has a discrepancy with regard to the target MGI object.
message	Populate with the discrepancy message. See “Associate Discrepancy Report Messages” on page 14.
_JobStream_key	Automatically populated by the Java Frameworks.
creation_date	Automatically populated by the Java Frameworks.

9.2 QC Reports

The association loader produces QC reports to show any inconsistencies encountered during the load. This is in addition to the validation errors that are written to the data validation log (See “Log Files” on page 16.). The Java application produces bcp files to load QC report tables with data that is needed for each report. After the load has completed, Python scripts are run to produce the reports using the data in the QC report tables. The following table lists the QC reports that are created by the association loader:

Table 10: QC Reports

Report Name	Report Table	Report File
Target Discrepancy Report	QC_AssocLoad_Target_Discrep	TargetDiscrepancy.rpt
Associate Discrepancy Report	QC_AssocLoad_Assoc_Discrep	AssocDiscrepancy.rpt

9.2.1 Target Discrepancy Report

This report shows any instances where the target accession ID/logical DB is not associated with one (and only one) MGI object of the expected type. The report contains the following information:

- Header - Copyright message, report date, database configuration, report title, column headings.
- Detail columns:
  - Acc ID - The target accession ID.

- Logical DB - The target logical DB.
- MGI Object - The MGI ID of the object that the target accession ID/logical DB is associated with (blank if no association exists).
- MGI Type - The MGI type of the object that the target accession ID/logical DB is associated with (blank if no association exists).
- Expected TypeD - The MGI type of the object that the target accession ID/logical DB is expected to be associated with.
- Message - A message that indicates why there is a discrepancy.

- Trailer - Total number of discrepancies.

9.2.1.1 Target Discrepancy Report Messages

The following table lists the messages that may appear in the Target Discrepancy Report. The message is prefixed with a letter (discrepancy code) to make it easier to identify particular messages in the report.

Table 11: Target Discrepancy Messages

Message	Reason
A: Same type (0), different type (0)	The target accession ID/logical DB is not associated with any MGI object.
B: Same type (0), different type (1)	The target accession ID/logical DB is only associated with one object of the wrong type.
C: Same type (0), different type (>1)	The target accession ID/logical DB is only associated with multiple objects of the wrong type.
D: Same type (1), different type (1)	The target accession ID/logical DB is associated with one object of the correct type, but also one object of the wrong type.
E: Same type (1), different type (>1)	The target accession ID/logical DB is associated with one object of the correct type, but also multiple objects of the wrong type.
F: Same type (>1), different type (0)	The target accession ID/logical DB is associated with multiple objects of the correct type.
G: Same type (>1), different type (1)	The target accession ID/logical DB is associated with multiple objects of the correct type and one object of the wrong type.

Table 11: Target Discrepancy Messages

Message	Reason
H: Same type (>1), different type (>1)	The target accession ID/logical DB is associated with multiple objects of the correct type and multiple objects of the wrong type.

9.2.2 Associate Discrepancy Report

This report shows any instances where an accession ID/logical DB is not associated with the target MGI object or they are associated with the wrong type or number of MGI objects. The report contains the following information:

- Header - Copyright message, report date, database configuration, report title, column headings.
- Detail columns:
  - Target Acc ID - The target accession ID.
  - Target Logical DB - The target logical DB.
  - Target MGI Object - The MGI ID of the object that the target accession ID/logical DB is associated with.
  - Target MGI Type - The MGI type of the object that the target accession ID/logical DB is associated with.
  - Acc ID - An accession ID that has a discrepancy with regard to its association (or lack thereof) with the target MGI object.
  - Logical DB - The logical DB for the accession ID.
  - MGI Object - The MGI ID of the object that the accession ID/logical DB is associated with.
  - MGI Type - The MGI type of the object that the accession ID/logical DB is associated with.
  - Message - A message that indicates why there is a discrepancy.
- Trailer - Total number of discrepancies.

9.2.2.1 Associate Discrepancy Report Messages

The following table lists the messages that may appear in the Target Discrepancy Report. The message is prefixed with a letter (discrepancy code) to make it easier to identify particular messages in the report.

Table 12: Associate Discrepancy Messages

Message	Reason
A: Same type (0), different type (1)	The accession ID/logical DB is only associated with one object of a different type than the target object.
B: Same type (0), different type (>1)	The accession ID/logical DB is associated with multiple objects of a different type than the target object.
C: Same type (1), same object (1), different type (1)	The accession ID/logical DB is associated with the target object, but also one object of the wrong type.
D: Same type (1), same object (1), different type (>1)	The accession ID/logical DB is associated with the target object, but also multiple objects of the wrong type.
E: Same type (1), same object (0), different type (0)	The accession ID/logical DB is associated with an object of the expected type, but it is not the target object.
F: Same type (1), same object (0), different type (1)	The accession ID/logical DB is associated with an object of the expected type, but it is not the target object. It is also associated with one object of a different type than the target object.
G: Same type (1), same object (0), different type (>1)	The accession ID/logical DB is associated with an object of the expected type, but it is not the target object. It is also associated with multiple objects of a different type than the target object.
H: Same type (>1), same object (1), different type (0)	The accession ID/logical DB is associated with multiple objects of the expected type, including the target object.

Table 12: Associate Discrepancy Messages

Message	Reason
I: Same type (>1), same object (1), different type (1)	The accession ID/logical DB is associated with multiple objects of the expected type, including the target object. It is also associated with one object of a different type than the target object.
J: Same type (>1), same object (1), different type (>1)	The accession ID/logical DB is associated with multiple objects of the expected type, including the target object. It is also associated with multiple objects of a different type than the target object.
K: Same type (>1), same object (0), different type (0)	The accession ID/logical DB is associated with multiple objects of the expected type, but not the target object.
L: Same type (>1), same object (0), different type (1)	The accession ID/logical DB is associated with multiple objects of the expected type, but not the target object. It is also associated with one object of a different type than the target object.
M: Same type (>1), same object (0), different type (>1)	The accession ID/logical DB is associated with multiple objects of the expected type, but not the target object. It is also associated with multiple objects of a different type than the target object.

9.3 Log Files

There are 4 log files that are written to by the association loader, as described in the following table:

Table 13: Log Files

Log Name	File Name
Process Summary Log	Defined by configuration variable LOG_PROC.
Diagnostic Log	Defined by configuration variable LOG_DIAG.
Curator Summary Log	Defined by configuration variable LOG_CUR.
Data Validation Log	Defined by configuration variable LOG_VAL.

See the DLA Standards document for a description of the information that is written to each of the log files.

10 Java Objects

10.1 MGI Association Object (MAO)

The association loader joins the *MGI_Association* table with the *ACC_Accession* table to identify all MGI objects that each accession ID/logical DB in the *MGI_Association* table are associated with. All rows of the results set with the same *_Record_key* are grouped together and used to set the attributes of a MGI association object (MAO). Once the MAO is loaded, it is handed off to the main algorithm to determine how to process it. The MAO contains the following attributes:

- The expected target type name that the target accession ID/logical DB should be associated with.
- The expected target type key that the target accession ID/logical DB should be associated with.
- A vector of accession IDs (one from each row of the results set).
- A vector of logical DB keys for the logical DB names for each accession ID.
- A vector of target indicators to indicate whether each accession ID represents the target MGI object or an accession ID to be associated with the target MGI object.
- A vector of MGI type keys for each MGI object that a correspond accession ID/logical DB is associated with (if any).
- A vector of object keys for each MGI object that a correspond accession ID/logical DB is associated with (if any).

11 Processing

11.1 Evaluating Accession IDs

The association loader needs to evaluate each of the accession ID/logical DB pairs in a MGI association object in order to determine how to process it. See "MGI Association Object (MAO)" on page 17. for more information about this object. Each MAO has one target accession ID/logical DB that is expected to be associated with one target MGI object. The MAO also has one or more non-target accession ID/logical DB pairs that the load will attempt to associate with the target object. The target accession ID/logical DB is evaluated first and if there are no discrepancies, the non-target accession ID/logical DB pairs are evaluated next.

11.1.1 Target Evaluation

The first step in evaluating a target accession ID/logical DB is to examine all of the MGI objects that it is associated with and generating the following counts:

- The number of associations to objects of the expected target type.
- The number of associations to objects of a different target type.

The next step is to determine which of the following actions to perform based on the counts:

C Continue - The target association is OK, so continue processing the other accession ID/logical DBs.

R,S Report and Skip - A target discrepancy exists, so report the discrepancy and end processing of this MAO.

The following table indicates which action to performed based on the counts. In the event of a discrepancy, the message code is also listed. See "Target Discrepancy Report" on page 12. for a description of the message codes:

Table 14: Action Evaluation For Target Accession ID/Logical DB

# of associations to objects with the expected target type	# of associations to objects with a different target type	Action Code	Message Code
0	0	R,S	A
0	1	R,S	B
0	> 1	R,S	C
1	0	C	N/A
1	1	R,S	D
1	> 1	R,S	E

Table 14: Action Evaluation For Target Accession ID/Logical DB

# of associations to objects with the expected target type	# of associations to objects with a different target type	Action Code	Message Code
> 1	0	R,S	F
> 1	1	R,S	G
> 1	> 1	R,S	G

11.1.2 Non-Target Evaluation

The first step in evaluating a non-target accession ID/logical DB is to examine all of the MGI objects that it is associated with and generating the following counts:

- The number of associations to objects with the same type as the target object
- The number of associations to the target object.
- The number of associations to objects with a different type than the target object.

The next step is to determine which action to perform. The action is determined by the counts and the logical DB itself. Each logical DB can be associated with either a single MGI object or multiple MGI objects. The logical DBs that fall into each category are defined by the configuration variables ASSOCLOAD_SINGLE_OBJECT_DB or ASSOCLOAD_MULTIPLE_OBJECT_DB. Based on the counts and the logical DB, one of the following actions it performed:

S Skip - The association already exists, so there is nothing to do.

R,S Report and Skip - A fatal discrepancy exists, so do not make any associations for this MAO.

A Associate - The association needs to be made to the target object.

R,A Report and Associate - A non-fatal discrepancy exists, so the association can still be made.

NOTE: If any non-target accession ID/logical DB is determined to have a “Report and Skip” action, no associations will be made for this MAO, regardless of how the other accession ID/logical DBs are evaluated.

The following tables indicate which action to performed based on the counts and the logical DB. The first table is used if the logical DB can only be associated with a single MGI object. The second table is used if the logical DB can be associated with multiple MGI objects. In the event of a discrepancy, the message code is also listed. See “Associate Discrepancy Report” on page 14. for a description of the message codes.

Table 15: Action Evaluation For Logical DBs Allowing Single Associations

# of associations to objects with the same type as the target object	# of associations to the target object	# of associations to objects with a different type than the target object	Action Code	Message Code
0	N/A	0	A	N/A
0	N/A	1	R,S	A
0	N/A	> 1	R,S	B
1	1	0	S	N/A
1	1	1	R,S	C
1	1	> 1	R,S	D
1	0	0	R,S	E
1	0	1	R,S	F
1	0	> 1	R,S	G
> 1	1	0	R,S	H
> 1	1	1	R,S	I
> 1	1	> 1	R,S	J
> 1	0	0	R,S	K
> 1	0	1	R,S	L
> 1	0	> 1	R,S	M

Table 16: Action Evaluation For Logical DBs Allowing Multiple Associations

# of associations to objects with the same type as the target object	# of associations to the target object	# of associations to objects with a different type than the target object	Action Code	Discrepancy Code
0	N/A	0	A	N/A
0	N/A	1	A	N/A

Table 16: Action Evaluation For Logical DBs Allowing Multiple Associations

# of associations to objects with the same type as the target object	# of associations to the target object	# of associations to objects with a different type than the target object	Action Code	Discrepancy Code
0	N/A	> 1	A	N/A
1	1	0	S	N/A
1	1	1	S	N/A
1	1	> 1	S	N/A
1	0	0	R,A	E
1	0	1	R,A	F
1	0	> 1	R,A	G
> 1	1	0	R,S	H
> 1	1	1	R,S	I
> 1	1	> 1	R,S	J
> 1	0	0	R,A	K
> 1	0	1	R,A	L
> 1	0	> 1	R,A	M

11.2 Association Algorithm

The association algorithm used by the association loader will process one MGI association object at a time (See “MGI Association Object (MAO)” on page 17. for more information about this object). The MAO contains all of the data necessary to evaluate the associations to be made. The algorithm follows these steps:

1. Perform the target evaluation on the target accession ID/logical DB to determine what action to perform. See “Target Evaluation” on page 18. for more information about this process.
2. If the action is “Report and Skip”, a target discrepancy needs to be reported. Write to the bcp file for the *QC_AssocLoad_Target_Discrep* table for each occurrence of the target accession ID/logical DB in the MAO. See “QC_AssocLoad_Target_Discrep.bcp” on page 10. for a description of the values that need to be written to the bcp file. **Do not process this MAO any further**.

3. For each non-target accession ID/logical DB, perform the non-target evaluation to determine what action to perform. See "Non-Target Evaluation" on page 19. for more information about this process.
4. For each non-target accession ID/logical DB, check the following:
   - 4.1 If the action is "Skip", the association already exists. **Continue with the next accession ID/logical DB**.
   - 4.2 If the action is "Report and Skip" or "Report and Associate", a discrepancy needs to be reported. Write to the bcp file for the *QC_AssocLoad_Assoc_Discrep* table for each occurrence of the non-target accession ID/logical DB in the MAO. See "QC_AssocLoad_Assoc_Discrep.bcp" on page 11. for a description of the values that need to be written to the bcp file.
   - 4.3 If the action is "Associate" or "Report and Associate" and none of the other accession ID/logical DB pairs has a "Report and Skip" action, the association needs to be made. Do the following to create the association:
     - 4.3.1. Write to the bcp file for the *ACC_Accession* table. See "ACC_Accession.bcp" on page 8. for a description of the values that need to be written to the bcp file.
     - 4.3.2. Write to the bcp file for the *ACC_AccessionReference* table. See "ACC_AccessionReference.bcp" on page 9. for a description of the values that need to be written to the bcp file.
     - 4.3.3. If the association is being made to a clone and the load reference has not been associated with this clone yet, write to the bcp file for the *PRB_Reference* table. See "PRB_Reference.bcp" on page 10. for a description of the values that need to be written to the bcp file.

11.3 Main Steps

This section describes the steps used by the association loader to process the associations defined in the RADAR database and get them into the MGD database. The processing involves the following steps:

1. Open the log files defined by the DLA Standards.
2. Check the configuration variable ASSOCLOAD_FROM_FILE to determine if the *MGI_Association* table needs to be loaded from an association input file. If this variable is set to "true", then do the following:
   - 2.1 Open the input file.
   - 2.2 Open a bcp file for the *MGI_Association* table.
   - 2.3 Read the header record from the input file to identify the logical DB for the target accession ID and the logical DB(s) for the accession ID(s) to be associated with the target.

2.4 Perform a lookup on each logical DB in the *ACC_LogicalDB* table. If any logical DB does not exist, write an error message to the data validation log and terminate the load.

2.5 For each of the remaining records in the input file (detail records), do the following:

2.5.1. Write to the bcp file for the *MGI_Association* table for the target accession ID in the first field of the input record. See "MGI_Association.bcp" on page 8. for a description of the values that need to be written to the bcp file.

2.5.2. Write to the bcp file for the *MGI_Association* table for each accession ID in the remaining fields of the input record. See "MGI_Association.bcp" on page 8. for a description of the values that need to be written to the bcp file.

2.6 Execute the bcp file to load the *MGI_Association* table.

3. Open bcp files for each of the tables that are needed for adding associations and references: *ACC_Accession*, *ACC_AccessionReference*, *PRB_Reference*.

4. Open bcp files for each of the tables that are needed for the QC reports: *QC_AssocLoad_Target_Discrep*, *QC_AssocLoad_Assoc_Discrep*.

5. Delete any existing records in the QC report tables using the following SQL:

```
TRUNCATE TABLE QC_AssocLoad_Target_Discrep
TRUNCATE TABLE QC_AssocLoad_Assoc_Discrep
```

6. If the delete/reload option is enabled, delete all records from the *ACC_Accession*, *ACC_AccessionReference* and *PRB_Reference* tables that were created by the job stream.

7. Find all of the MGI objects that the accession IDs in the *MGI_Association* table are associated with. All row of the results set with the same *_Record_key* will be used to load a MGI association object (MAO). See "MGI Association Object (MAO)" on page 17.for more information about this object. Use the following SQL to generate the results set:

```
SELECT m._Record_key,
       m.accID,
       db._LogicalDB_key,
       m.target,
       null 'MGI Type',
       null 'Object Key'
FROM radar..MGI_Association m,
     mgd..ACC_LogicalDB db
WHERE m._JobStream_key = [JOBKEY system property] and
      m.logicalDB = db.name and
      not exists (SELECT 1
```

```
                    FROM mgd..ACC_Accession a,
                         mgd..ACC_LogicalDB db2
                    WHERE m.accID = a.accID and
                          m.logicalDB = db2.name and
                          db2._LogicalDB_key = a._LogicalDB_key)
UNION
SELECT m._Record_key,
       m.accID,
       db._LogicalDB_key,
       m.target,
       a._MGIType_key 'MGI Type',
       a._Object_key 'Object Key'
FROM radar..MGI_Association m,
     mgd..ACC_Accession a,
     mgd..ACC_LogicalDB db
WHERE m._JobStream_key = [JOBKEY system property] and
      m.accID = a.accID and
      m.logicalDB = db.name and
      db._LogicalDB_key = a._LogicalDB_key and
      a._MGIType_key != 21
ORDER BY m._Record_key, m.accID, db._LogicalDB_key
```

NOTE: This query excludes any associations to Nomenclature (MGI Type = 21).

8. For each MAO generated from the results set, do the following:
   8.1 Process the MAO according to the Association Algorithm See “Association Algorithm” on page 21. for more information.
9. Execute the bcp files to load the *ACC_Accession*, *ACC_AccessionReference* and *PRB_Probe* tables.
10. Execute the bcp files to load the *QC_AssocLoad_Target_Discrep* and *QC_AssocLoad_Assoc_Discrep* tables.
11. Write the processing counts to the curator summary log.
12. Close the log files.

12 Exceptions

Any exception that occurs during processing is considered to be fatal. This will cause the association loader to terminate immediately. The following conditions will cause exceptions in the association loader:

- An output file (bcp or log) cannot be written to.
- The bcp process does not complete successfully.
- An SQL error occurs.
- A Java object cannot be instantiated.