INTERNATIONAL TRANSACTION LOG

DRAFT
TECHNICAL DESIGN SPECIFICATION

ANNEXES

Non-paper

7 June, 2004

Table of Contents

Annex A
Glossary of Terms

[to be completed]

Annex B
Database Entity Relationship Diagrams

1. Overview

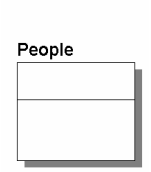
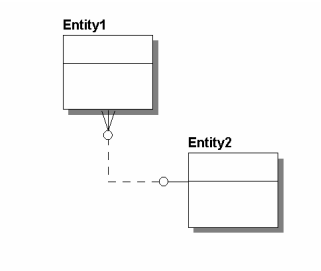
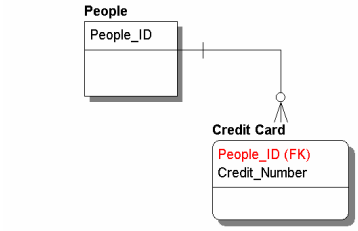
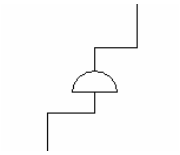
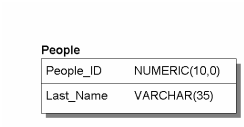
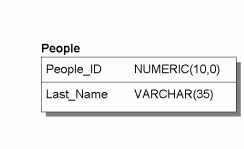
The data model presented in this annex represents a logical view of the database design for the ITL. It does not include the entities needed for the ITL AA, which might include additional report and job catalogs.

This Entity Relationship Diagram (ERD) and all its submodels are logical data models, not physical database design. Therefore, these models are normalised and may require some denormalisation when designing the physical database. These logical data models represent all of the possible data requirements for the ITL. They do not include the business processing rules that supplement the data model, which can be found in Annex E.

1.1 Conventions and Standards

The following figure describes the naming standards and diagramming conventions used in these models.

Figure B1: Model Conventions and Standards

Element	Name	Description
	Entity	An entity represents a set of real or abstract things (people, places, events) that have common attributes or characteristics. Entities may be either independent or dependent and are the logical equivalent to a table. An independent entity is called a "parent" and the dependent tables to it are "children."
	Relationship	Relationships define how two entities are associated with each other. Relationships may be either optional or mandatory. A dotted line means the relationship is optional and a continuous line means that the relationship is mandatory.
	Cardinality	The small circle at the end of a relationship line shows the ratio of parent entity to child entities. The presence of the circular dot defines where there is more than one occurrence of a dependent entity. For example, for each person (in the People entity) there are at least one or more Credit Cards associated with that person.
	Subtype	A subtype is a relationship in which information is stored about a specific type of some other entity. For example, a SALARIED EMPLOYEE is a specific type of EMPLOYEE. Subtypes are used for expressing relationships that are only valid for that specific subtype.
	Attributes	An attribute represents a type of characteristic or property about an entity. An attribute is described as having a datatype and length. Datatypes can be numeric, date and/or time, or variable character (Varchar).
	Primary Key	When the attribute uniquely defines a single set of elements within an entity, that attribute (or set of attributes) is referred to as a Primary Key. Primary keys are shown at the top of the entity above the dividing line which distinguishes it from non-primary attributes.

1.2 Overview Model and Submodels

Due to the size of the ERD, the model is divided into subsections. The overview model (Figure B2) is an entity-only view of all entities and their relationships. For the purpose of clarity, all lookup tables which serve as foreign keys in the child tables have been removed from this model. Subsequent submodels may show relevant lookup tables as space provides. The remaining submodels provide attribute, datatype, and cardinality information, supporting the major processing types as defined in Section 3.5 of the International Transaction Log Technical Design Specification.

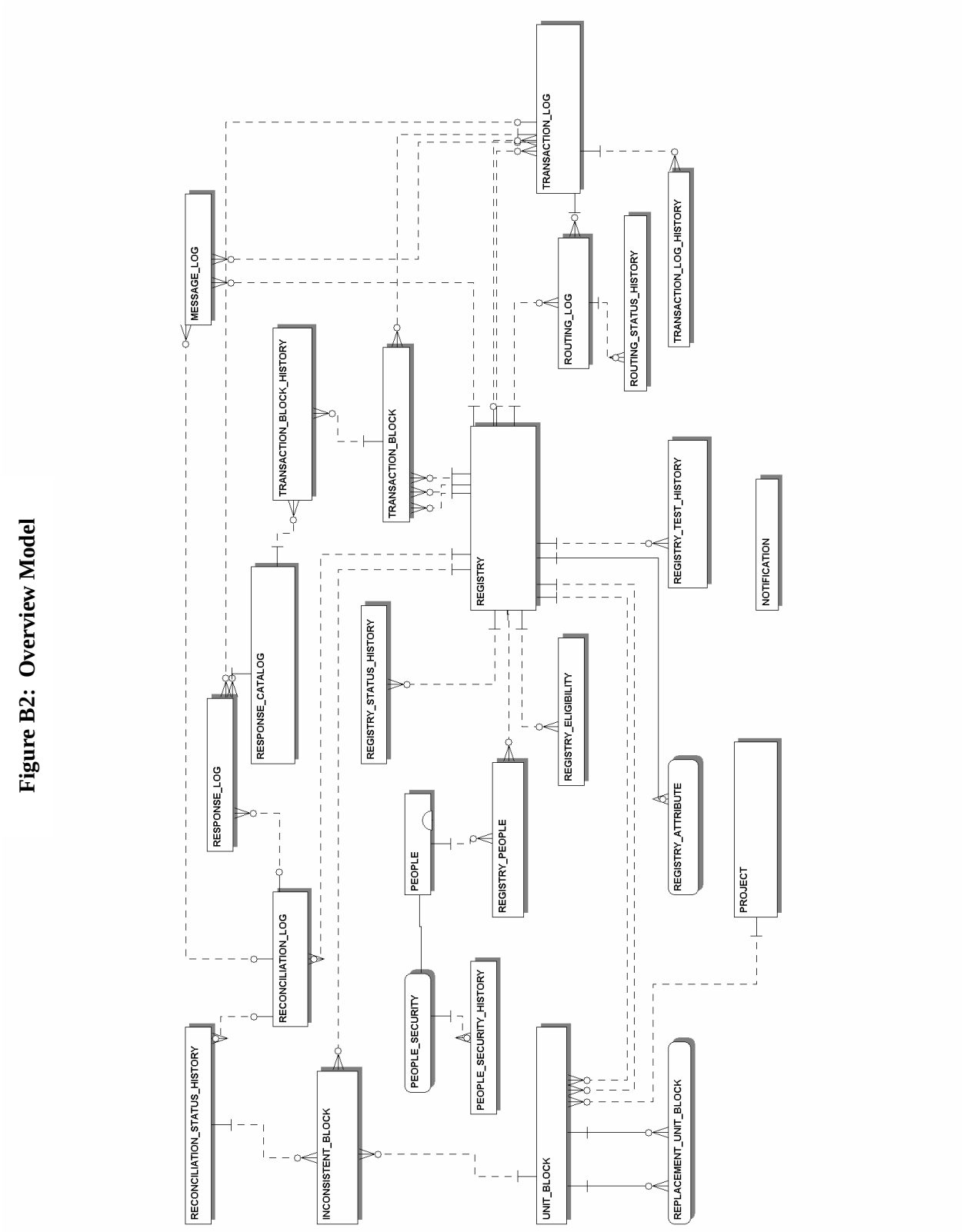


Figure B2: Overview Model

Figure B3: Registry Submodel

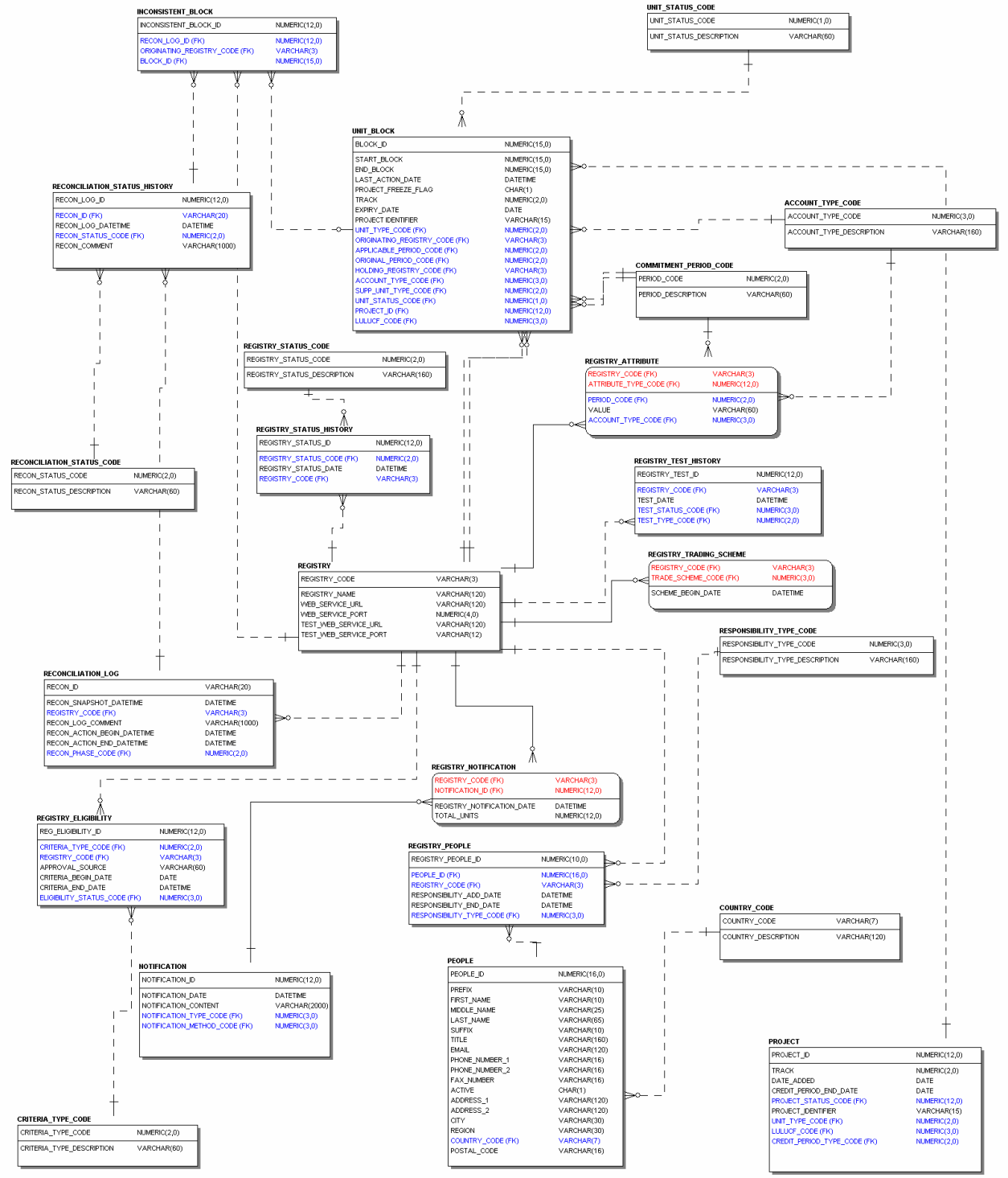


Figure B4: Transaction Process Submodel

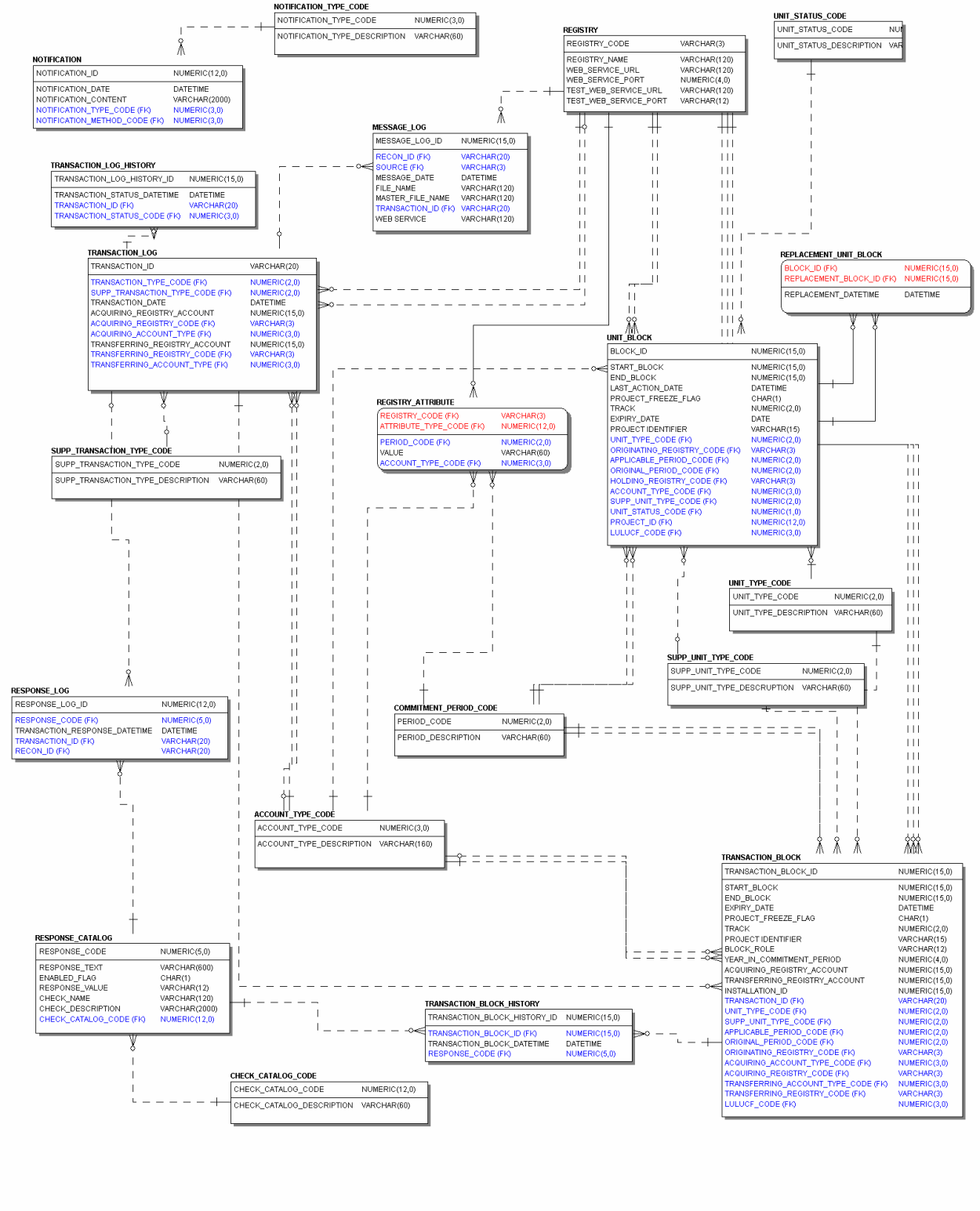


Figure B5: Reconciliation Submodel

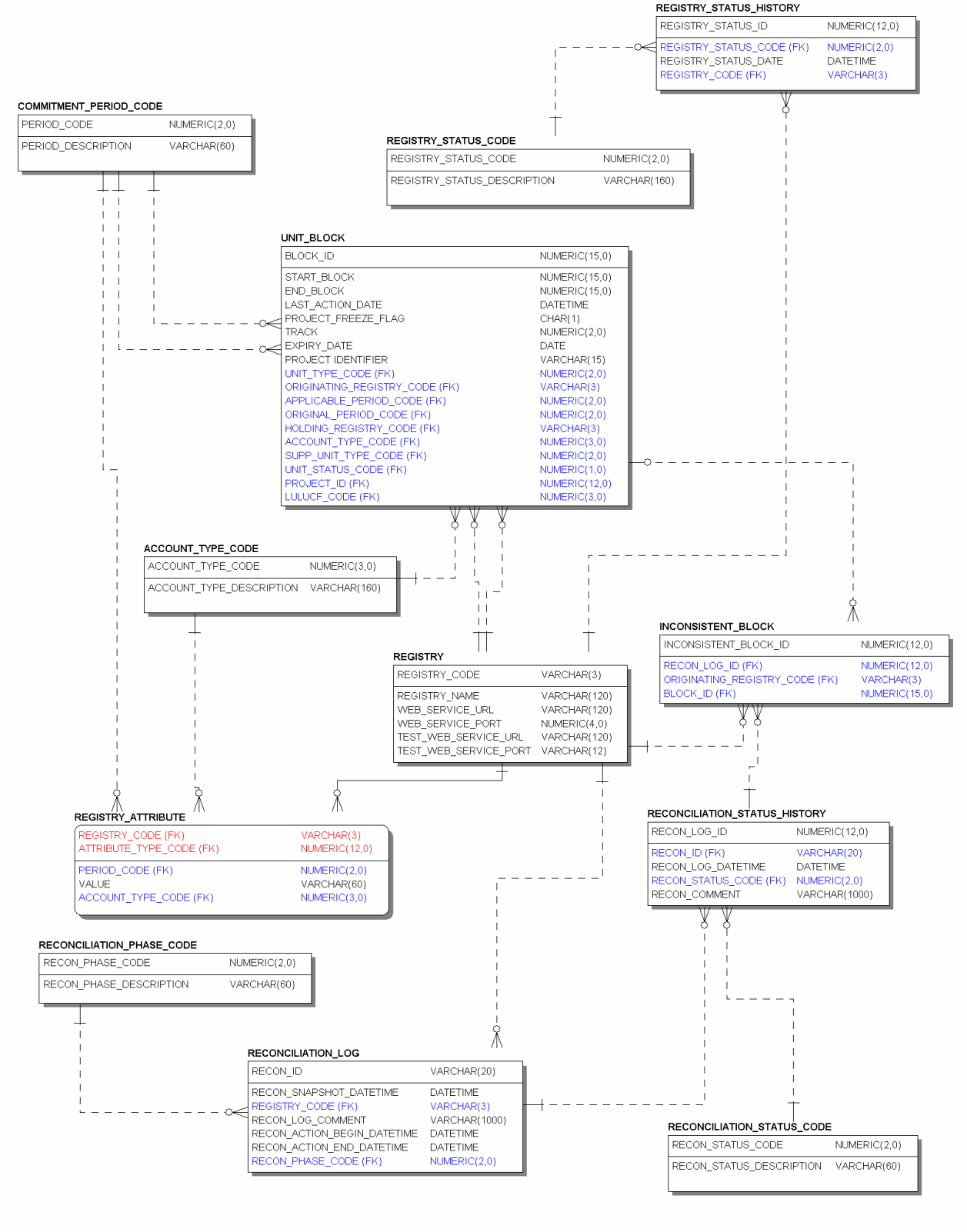


Figure B6: System Data Submodel

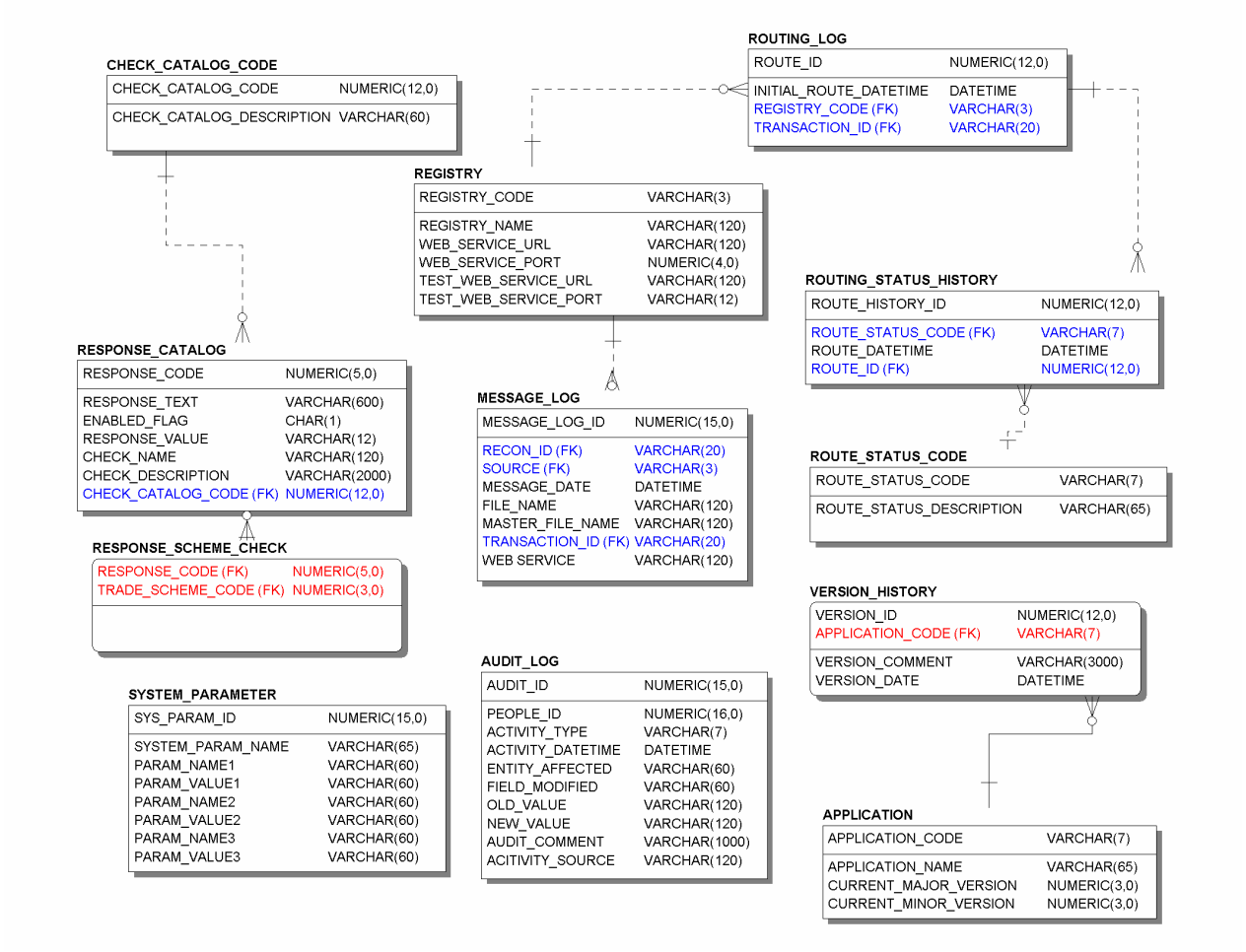
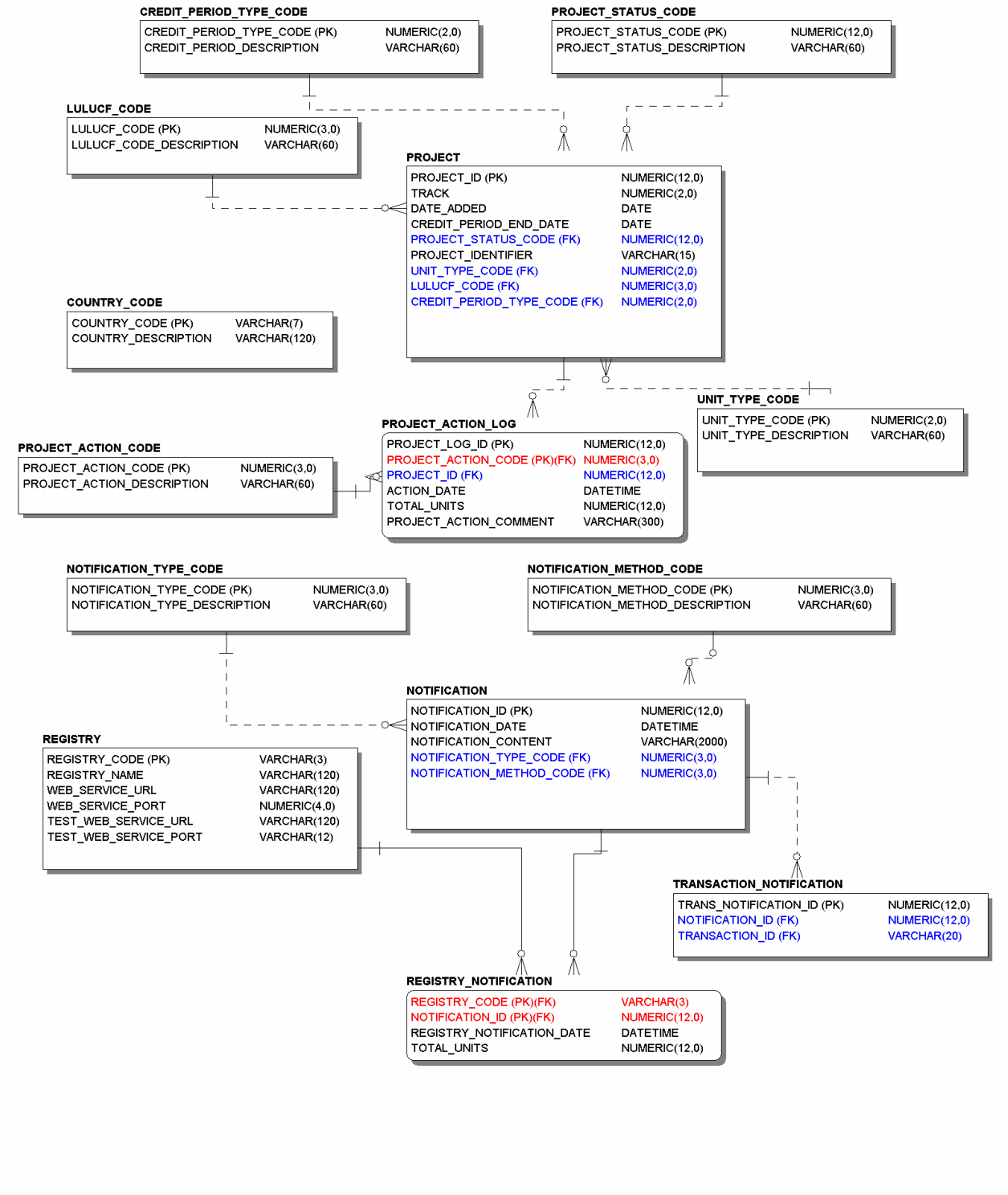


Figure B7: Projects and Notifications Submodel



Annex C
Data Element Definitions

Figure C1: List of Entities

Entity Name	Description
Account Type Code	Lookup table of account types. Reference Annex D, Figure D2.
Application	Contains information about the current versions of ITL applications and related technical requirements (i.e., ITL AA, ITL PubWeb, DES, STL).
Attribute Type Code	Attributes describe various traits of a registry. Typical codes identify a registry's reserve, and initial allocation.
Audit Log	Contains record of all database actions occurring on selected tables in the database (i.e., inserts, updates and deletes).
Check Catalog Code	Collection of Rule Checks performed on transactions. These checks correspond to response codes for both success or failure on a check.
Commitment Period Code	Lookup table of Commitment Periods. Reference Annex D, Figure D5.
Country Code	Lookup table of country codes per ISO3166.
CreditingPeriod Type Code	Lookup table of crediting period type codes.
Criteria Type Code	Lookup table of eligibility constraints.
Eligibility Status Code	Lookup table of eligibility status codes.
Inconsistent Block	Contains unit blocks that are identified during reconciliation action as potentially inconsistent.
LULUCF Code	Lookup table of LULUCF activity codes. Reference Annex D, Figure D9.
Message Log	Records all incoming and outgoing transaction messages as received from or sent to national registries.
Notification	Table of notifications that can be sent to registries.
Notification Method Code	Lookup table of methods of how a notification was sent.
Notification Type Code	Lookup table of notification type codes.
People	Contains information on people that have a relationship with either the ITL, STL, or a registry.
People Security	Contains User IDs and current password providing access to the ITL.

(cont.)

Figure C1: List of Entities (cont.)

Entity Name	Description
People Security History	Contains history of previously used passwords for a person assigned a User ID.
Project	Table of Projects that are managed by the CDM.
Project Action Code	Lookup table of Project Action Codes.
Project Action Log	History table of all the actions applied to a Project.
Project Status Code	Lookup table of Project status codes.
Reconciliation Log	Contains information about each reconciliation action initiated by the ITL, including date and time and comments.
Reconciliation Phase Code	Lookup table of reconciliation phase codes.
Reconciliation Status Code	Lookup table of reconciliation status codes. Reference Annex D, Figure D16.
Reconciliation Status History	Contains information about each stage of the reconciliation action, including the status, date and time, and comments.
Registry	Registries with whom the ITL communicates to validate and process transactions.
Registry Attribute	Table that contains information regarding a registry's characteristics such as its initial allocation or current reserve.
Registry Eligibility	Table of eligibility requirements met by a registry. Managed by the CNA data system.
Registry Notification	Table of all notifications sent to a registry.
Registry People	Relationship between a person and a registry.
Registry Status Code	Lookup table of status codes for a registry, identifying whether it is eligible to communicate with the ITL/STL. Reference Annex D, Figure D17.
Registry Status History	Contains a record of each of the various registry status changes.
Registry Test History	History of registry tests conducted.
Registry Trading Scheme	Contains information about additional trading programs.
Replacement Unit Block	Contains information about unit blocks that have been replaced (as in tCERs and lCERs). There can only be one replacement record for a unit block.

(cont.)

Figure C1: List of Entities (cont.)

Entity Name	Description
Response Catalog	Lookup table of all response codes per DES Annex C. Contains indicator that response code is active and the response type. (Also contains CITL response codes.)
Response Log	Contains, for each transaction or reconciliation, all response codes generated by the ITL in its evaluation. These responses are non-unit block specific.
Response Scheme Check	Table of checks that are specific for a particular trading scheme.
Responsibility Type Code	Lookup table of responsibility codes that describe a person's responsibilities for a registry. Reference Annex D, Figure D18.
Route Status Code	Lookup table of codes describing routing log action. Reference Annex D, Figure D19.
Routing Log	Log of messages sent to and from the STL.
Routing Status History	Contains, for each transaction, a record of each status change sent to or received from the STL.
Supplementary Transaction Type Code	Lookup table of STL supplemental transaction type codes.
Supplementary Unit Type Code	Lookup table of STL supplemental unit type codes.
System Log	Contains information on administrative processes.
System Parameter	Contains system variables, to be defined as needed, for system configuration or system management.
Test Status Code	Lookup table of test results or test status. References Annex D, Figure D20.
Test Type Code	Lookup table of tests (to be defined).
Trade Scheme Code	Lookup table of trade schemes.
Transaction Block	Identifies the unit block(s) that are part of each transaction.
Transaction Block History	Contains, for each transaction, all response codes associated with a specific unit block.
Transaction Log	Contains a single record of each transaction that is submitted for validation to the ITL.
Transaction Log History	Contains a record of each stage and status associated with processing a specific transaction.
Transaction Notification	Contains records of all transactions that were proposed in compliance with a notification.

(cont.)

Figure C1: List of Entities (cont.)

Entity Name	Description
Transaction Status Code	Lookup table of transaction status codes. Reference Annex D, Figure D23.
Transaction Type Code	Lookup table of transaction types. Reference Annex D, Figure D24.
Unit Block	Contains all of the units (defined by start and end blocks) which have been issued by registries under the Kyoto Protocol.
Unit Status Code	Lookup table of reasons a unit block may be in holding queue. Reference Annex D, Figure D25.
Unit Type Code	Lookup table of unit (allowance) types. Reference Annex D, Figure D26.
Version History	Contains information about versions of the DES.

58
59

Figure C2: Account Type Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Account Type Code	Number(3)	N	Y	Numeric code indicating the type of account.
Account Type Description	Varchar2(160)	Y	N	Text description of an account type.

60

Figure C3: Application Entity Details

Attribute	Data Type	Null	Primary Key	Description
Application Code	Varchar2(7)	N	Y	Primary key code for application.
Application Name	Varchar2(65)	Y	N	Name of application.
Current Major Version	Number(3)	Y	N	Current major version of application.
Current Minor Version	Number(3)	Y	N	Current minor version of application.

61
62

Figure C4: Attribute Type Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Attribute Type Code	Number(12)	N	Y	Unique identifier of attribute type code.
Attribute Type Description	Varchar2(60)	Y	N	Description of attribute type code.

63
64

Figure C5: Audit Log Entity Details

Attribute	Data Type	Null	Primary Key	Description
Audit ID	Number(15)	N	Y	Unique identifier for the Audit_Log table.
Activity Type	Varchar2(7)	N	N	The activity or action (often a table insert, update, or delete) for which an audit record is made.
Activity DateTime	DateTime	N	N	The date and time of an activity.
Entity Affected	Varchar2(60)	N	N	The table which was modified by an activity.
Field Modified	Varchar2(60)	N	N	The table field that was modified by an activity.
Old Value	Varchar2(120)	N	N	The value of a field prior to an activity.
New Value	Varchar2(120)	N	N	The value of a field after an activity has modified the field.
Audit Comment	Varchar2(1000)	Y	N	Additional comments regarding an activity.
Activity Source	Varchar2(120)	Y	N	Identifies where the transaction originated from, generally as a terminal identification name or number.
People ID	Number(16)	Y	N	Unique identifier for a person.

65
66

Figure C6: Check Catalog Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Check Catalog Code	Numeric(12)	N	Y	Unique identifier of check group.
Check Catalog Description	Varchar2(60)	Y	N	Description of check group.

67
68

Figure C7: Commitment Period Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Period Code	Number(2)	N	Y	Code indicating the Commitment Period block.
Period Description	Varchar2(60)	Y	N	Text description of a Commitment Period.

Figure C8: Country Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Country Code	Varchar2(7)	N	Y	Code indicating a country.
Country Description	Varchar2(120)	Y	N	Text description of a country.

Figure C9: Credit Period Type Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Credit Period Type Code	Number(2)	N	Y	Unique identifier for credit period type code.
Credit Period Description	Varchar2(60)	Y	N	Description of credit period type code.

Figure C10: Criteria Type Code

Attribute	Data Type	Null	Primary Key	Description
Criteria Type Code	Number(2)	N	Y	Unique identifier of a criteria type code.
Criteria Type Description	Varchar2(60)	Y	N	Description of criteria type code.

Figure C11: Eligibility Status Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Eligibility Status Code	Number(3)	N	Y	Unique identifier of an eligibility status code.
Eligibility Status Description	Varchar2(60)	Y	N	Text description of an eligibility status code.

77
78

Figure C12: Inconsistent Block Entity Details

Attribute	Data Type	Null	Primary Key	Description
Inconsistent Block ID	Number(12)	N	Y	Reference to a unit block involved in a reconciliation action.
Recon Log ID	Number(12)	N	N	Unique identifier of a record in the Reconciliation_Log table.
Originating Registry Code	Varchar2(3)	N	N	Code identifying the registry of the block.
Block ID	Number(15)	N	N	Unique identifier of a unit block.

79
80

Figure C13: LULUCF Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
LULUCF Code	Number(3)	N	Y	Code indicating the LULUCF activity associated with the units in the block.
LULUCF Code Description	Varchar2(60)	Y	N	Text description of a LULUCF activity.

81
82

Figure C14: Message Log Entity Details

Attribute	Data Type	Null	Primary Key	Description
Message Log ID	Number(15)	N	Y	Unique identifier for the Message_Log table.
Message Date	DateTime	N	N	Date and time an incoming message was received.
Master File Name	Varchar2(120)	Y	N	Name of Zipped master file that was created when incoming message was parsed to an XML document.
File Name	Varchar2(120)	Y	N	The file in which a message is stored. Different transaction types may be stored in separate paths.
Transaction ID	Varchar2(20)	Y	N	Unique identifier for a transaction.
Recon ID	Varchar2(20)	Y	N	Unique identifier for a reconciliation action.
Web Service	Varchar2(120)	Y	N	Web service involved in sending or receiving messages.
Source	Varchar2(3)	N	N	Code identifying the registry or STL as source of message.

Figure C15: Notification Entity Details

Attribute	Data Type	Null	Primary Key	Description
Notification ID	Number(12)	N	Y	Unique identifier of a notification record.
Notification Date	DateTime	Y	N	Date and time a notification was created.
Notification Content	Varchar2(2000)	Y	N	Content notification message.
Notification Type Code	Number(3)	N	N	Unique identifier of notification type code.
Notification Method Code	Number(3)	Y	N	Unique identifier of notification method code.
Project Log ID	Number(12)	Y	N	Unique identifier of Project action log record.
Parent Notification ID	Number(12)	Y	N	Identifies the Parent notification of Zero indicates a parent notification.

Figure C16: Notification Method Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Notification Method Code	Number(3)	N	Y	Unique identifier of notification method code.
Notification Method Description	Varchar2(60)	Y	N	Description of notification method code.

Figure C17: Notification Type Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Notification Type Code	Number(3)	N	Y	Unique identifier of notification type codes.
Notification Type Description	Varchar2(60)	Y	N	Description of notification type codes.

Figure C18: People Entity Details

Attribute	Data Type	Null	Primary Key	Description
People ID	Number(16)	N	Y	Unique identifier for a person.
Prefix	Varchar2(10)	Y	N	The prefix of the person (Mr., Dr., etc.).
First Name	Varchar2(10)	Y	N	The first name or given name of a person.
Middle Name	Varchar2(25)	Y	N	The middle name of a person.
Last Name	Varchar2(65)	N	N	The last name or family name of a person.
Suffix	Varchar2(10)	Y	N	The suffix on a person's name, such as Jr.
Title	Varchar2(160)	Y	N	The title by which a person is referred, such as Mr. or Mrs.
Email	Varchar2(120)	Y	N	The email address of a person.
Phone Number 1	Varchar2(16)	Y	N	Primary phone number.
Phone Number 2	Varchar2(16)	Y	N	Secondary or mobile phone number.
Fax Number	Varchar2(16)	Y	N	The fax number of a person.
Active	Char(1)	Y	N	Indicator of whether or not the person is active. By default, a null value indicates active, a 1 indicates a non-active person.

(cont.)

Figure C18: People Entity Details (cont.)

Attribute	Data Type	Null	Primary Key	Description
Address 1	Varchar2(120)	Y	N	The first line of an address for a person.
Address 2	Varchar2(120)	Y	N	The second line of an address for a person.
City	Varchar2(30)	Y	N	The city in an address for a person.
Region	Varchar2(30)	Y	N	The region in which the address of a person is located.
Country Code	Varchar2(7)	N	N	Code indicating a country for the person's address.
Postal Code	Varchar2(16)	Y	N	The post code for the address of a person.

87
88

Figure C19: People Security Entity Details

Attribute	Data Type	Null	Primary Key	Description
People ID	Number(16)	N	Y	Unique identifier for a person.
Password	Varchar2(120)	Y	N	The password of a person.
Password Change Date	DateTime	Y	N	The date and time a person’s password was last changed.
Login	Varchar2(12)	Y	N	The User ID assigned to a person.
Security Role	Number(1)	N	N	Security role access to the ITLAA. 1 = Audition, Z = Registry Administrator, 3 = System Administrator

89
90

Figure C20: People Security History Entity Details

Attribute	Data Type	Null	Primary Key	Description
Security ID	Number(6)	N	Y	Identity key for Password_History table.
Change Date	DateTime	Y	N	Date and time a password was changed.
Password	Varchar2(120)	Y	N	The old encrypted password of a person.
People ID	Number(16)	N	N	Unique identifier for a person.
Security Role	Number(1)	N	N	Security role access to the ITLAA. 1 = Audition, Z = Registry Administrator, 3 = System Administrator

91

Figure C21: Project Entity Details

Attribute	Data Type	Null	Primary Key	Description
Project ID	Number(12)	N	Y	Unique identifier of a Project.
Track	Number(2)	Y	N	Code identifying the Project track.
Date Added	Date	Y	N	Date when the Project was added.
Crediting Period End Date	Date	Y	N	Date in which Project expires. After this date, lCERs for the Project can no longer be used for compliance and must be updated.
Unit Type Code	Number(2)	N	N	Code indicating the type of units in a block.
Project Status Code	Number(12)	N	N	Unique identifier of Project status codes.
LULUCF Code	Number(3)	N	N	Code indicating the LULUCF activity associated with the units in the block.
Crediting Period Type Code	Number(12)	N	N	Unique identifier for crediting period type code.
Project Identifier	Varchar2(12)	Y	N	Country code and Project number which uniquely identifies a JI Project.

92
93

Figure C22: Project Action Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Project Action Code	Number(3)	N	Y	Unique identifier of a Project action code.
Project Action Description	Varchar2(60)	Y	N	Description of a Project action code.

94
95

Figure C23: Project Action Log Entity Details

Attribute	Data Type	Null	Primary Key	Description
Project Log ID	Number(12)	N	Y	Unique identifier of Project action log record.
Project ID	Number(12)	N	N	Unique identifier of a Project assigned by a CDM.
Action Date	DateTime	Y	N	Date in which action occurred.
Total Units	Number(12)	Y	N	Total number of units affected by this Project action.
Project Action Comment	Varchar2(300)	Y	N	Comment on Project action.
Project Action Code	Number(3)	N	N	Unique identifier of a Project action code.

96
97

Figure C24: Project Status Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Project Status Code	Number(12)	N	Y	Unique identifier of Project status codes.
Project Status Description	Varchar2(60)	Y	N	Description of Project status code.

98
99

Figure C25: Reconciliation Log Entity Details

Attribute	Data Type	Null	Primary Key	Description
Recon ID	Varchar2(20)	N	Y	Unique identifier for a reconciliation action.
Recon Log Comment	Varchar2(1000)	Y	N	Comment regarding a reconciliation action.
Registry Code	Varchar2(3)	N	N	Code identifying the registry associated with a reconciliation action.
Recon Action Begin DateTime	DateTime	N	N	Date and time a reconciliation action was initiated by the ITL.
Recon Action End DateTime	DateTime	N	N	Date and time a reconciliation action was concluded by the ITL.
Recon Snapshot DateTime	DateTime	N	N	Date and time for which snapshot is to be provided.
Recon Phase Code	Number(2)	N	N	Unique identifier for a reconciliation start phase.

100
101

Figure C26: Reconciliation Phase Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Recon Phase Code	Number(2)	N	Y	Unique identifier of a reconciliation phase code.
Recon Phase Code Description	Varchar2(60)	Y	N	Description of a reconciliation phase code.

102
103

Figure C27: Reconciliation Status Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Recon Status Code	Number(2)	N	Y	Code indicating the status of a reconciliation action.
Recon Status Description	Varchar2(60)	Y	N	Text description of a status for a reconciliation action.

104
105

Figure C28: Reconciliation Status History Entity Details

Attribute	Data Type	Null	Primary Key	Description
Recon Log ID	Number(12)	N	Y	Unique identifier of a record in the Reconciliation_Log table.
Recon ID	Varchar2(20)	N	N	Unique Identifier for a reconciliation action.
Recon Log DateTime	DateTime	Y	N	Date and time a record is made in the reconciliation status history.
Recon Status Code	Number(2)	N	N	Code indicating the status of a reconciliation action.
Recon Comment	Varchar2(1000)	Y	N	Comment regarding the status of a reconciliation process.

106
107

Figure C29: Registry Entity Details

Attribute	Data Type	Null	Primary Key	Description
Registry Code	Varchar2(3)	N	Y	Code identifying the registry.
Registry Name	Varchar2(120)	Y	N	Name of a registry.
Web Service URL	Varchar2(120)	Y	N	The Web service URL from which the registry will broadcast its Web services.
Web Service Port	Number(4)	Y	N	The port on which the registry will run its Web services.
Test Web Service URL	Varchar2(120)	Y	N	The Web service URL from which the registry will broadcast its test Web services.
Test Web Service Port	Number(4)	Y	N	The port on which the registry will run its test Web services.

108
109

Figure C30: Registry Attribute Entity Details

Attribute	Data Type	Null	Primary Key	Description
Registry Code	Varchar2(3)	N	Y	Code identifying the registry.
Period Code	Number(2)	N	Y	Code indicating the Commitment Period.
Value	Number(12)	N	N	Identifies the value for the attribute specific to the registry.
Attribute Type Code	Number(3)	N	N	Numeric code indicating type of attribute.

110
111

Figure C31: Registry Eligibility Entity Details

Attribute	Data Type	Null	Primary Key	Description
Reg Eligibility ID	Number(12)	N	Y	Unique identifier for a registry eligibility record.
Criteria Begin Date	Date	Y	N	Date in which eligibility action occurred.
Approval Source	Varchar2(60)	Y	N	Identifies who approved the eligibility requirement.
Registry Code	Varchar2(3)	N	N	Code identifying the registry.
Criteria Type Code	Number(2)	N	N	Unique identifier of eligibility type code.
Criteria End Date	DateTime	Y	N	Date when eligibility action for registry became invalid.
Eligibility Status Code	Number(3)	N	N	Unique identifier of an eligibility status code.

112
113

Figure C32: Registry Notification Entity Details

Attribute	Data Type	Null	Primary Key	Description
Registry Code	Varchar2(3)	N	Y	Code identifying the registry.
Notification ID	Number(12)	N	Y	Unique identifier of a notification record.
Total Units	Number(12)	Y	N	Total number of units affected by the Project action for this registry.
Registry Notification Date	DateTime	Y	N	Date and time in which notification was sent.

114
115

Figure C33: Registry People Entity Details

Attribute	Data Type	Null	Primary Key	Description
Registry People ID	Number(10)	N	Y	Unique identifier for a registry-person relationship.
People ID	Number(16)	N	N	Unique identifier for a person.
Registry Code	Varchar2(3)	N	N	Code identifying the registry.
Responsibility Add Date	DateTime	N	N	The date on which a registry-people responsibility started.
Responsibility End Date	DateTime	Y	N	The date on which a registry-people responsibility ended.
Responsibility Type Code	Number(3)	N	N	Code indicating the type of responsibility that exists between a registry and a person.

116
117

Figure C34: Registry Status Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Registry Status Code	Number(2)	N	Y	Code indicating an operating or permission status for registries.
Registry Status Description	Varchar2(160)	Y	N	Text description of a registry status.

118
119

Figure C35: Registry Status History Entity Details

Attribute	Data Type	Null	Primary Key	Description
Registry Status ID	Number(12)	N	Y	Unique identifier for a registry status record.
Registry Status Code	Number(2)	N	N	Code indicating an operating or permissions status for registries.
Registry Status Date	DateTime	Y	N	Date and time on which status began.
Registry Code	Varchar2(3)	N	N	Code identifying the registry.

120
121

Figure C36: Registry Test History Entity Details

Attribute	Data Type	Null	Primary Key	Description
Registry Code	Varchar2(3)	N	Y	Code identifying a registry.
Test Date	DateTime	Y	N	Date on which a test was received, performed or completed.
Test Status Code	Number(3)	N	N	Code indicating the status of a test or test submission.
Test Type Code	Number(2)	N	N	Code indicating type of test submission.

122
123

Figure C37: Registry Trading Scheme Entity Details

Attribute	Data Type	Null	Primary Key	Description
Registry Code	Varchar2(3)	N	Y	Code identifying a registry.
Trade Scheme Code	Number(3)	N	Y	Code identifying trade scheme.
Scheme Begin Date	DateTime	N	N	Date registry began participating in trading scheme.

124
125

Figure C38: Replacement Unit Block Entity Details

Attribute	Data Type	Null	Primary Key	Description
Block ID	Number(15)	N	Y	Unique identifier of a unit block.
Replacement Block ID	Number(15)	N	Y	Unique identifier of a unit block that was replaced.
Replacement DateTime	DateTime	N	N	Date and time when unit block was replaced.

126

Figure C39: Response Catalog Entity Details

Attribute	Data Type	Null	Primary Key	Description
Response Code	Number(5)	N	Y	Code indicating a response to a transaction process, such as a transaction validation.
Response Text	Varchar2(600)	Y	N	The English text describing a response.
Enabled Flag	Char(1)	Y	N	Identifies whether the response code is currently enabled or not.
Response Value	Varchar2(12)	Y	N	Identifies if the response is for a success or failure on a check.
Check Name	Varchar2(120)	Y	N	Name of rule or check.
Check Description	Varchar2(2000)	Y	N	Text description of a check.
Check Catalog Code	Number(12)	N	N	Unique identifier of check group.

Figure C40: Response Log Entity Details

Attribute	Data Type	Null	Primary Key	Description
Response Code	Number(5)	N	N	Code indicating a response to a transaction process, such as a transaction validation.
Transaction ID	Varchar2(20)	Y	N	Unique identifier for a transaction.
Transaction Response DateTime	DateTime	N	N	Date and time a transaction response is sent.
Response Log ID	Number(12)	N	Y	Unique identifier for a Response_Log record.
Recon ID	Varchar2(20)	Y	N	Unique identifier for a reconciliation action.

Figure C41: Response Scheme Check Entity Details

Attribute	Data Type	Null	Primary Key	Description
Response Code	Number(5)	N	Y	Code indicating a response to a transaction process, such as a transaction validation.
Trade Scheme Code	Number(3)	N	Y	Unique identifier of a trading scheme code.

Figure C42: Responsibility Type Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Responsibility Type Code	Number(3)	N	Y	Code indicating the type of relationship between a registry and a person.
Responsibility Type Description	Varchar2(160)	Y	N	Text description of the relationship between a person and a registry.

133
134

Figure C43: Route Status Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Route Status Code	Varchar2(7)	N	Y	Code indicating the status of a routing event.
Route Status Description	Varchar2(65)	Y	N	Text description of a routing event status.

135
136

Figure C44: Routing Log Entity Details

Attribute	Data Type	Null	Primary Key	Description
Route ID	Number(12)	N	Y	Unique identifier for a routing event.
Initial Route DateTime	DateTime	Y	N	Date and time of a routing event.
Registry Code	Varchar2(3)	N	N	Code identifying a registry.
Transaction ID	Varchar2(20)	N	N	Unique identifier for a transaction.

137
138

Figure C45: Routing Status History Entity Details

Attribute	Data Type	Null	Primary Key	Description
Route History ID	Number(12)	N	Y	Unique identifier for each status change of a routing event.
Route Status Code	Varchar2(7)	N	N	Code indicating the status of a routing event.
Route DateTime	DateTime	Y	N	Date and time of a routing event.
Route ID	Number(12)	N	N	Unique identifier for a routing event.

139

Figure C46: Supplementary Transaction Type Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Supplementary Transaction Type Code	Number(2)	N	Y	Code indicating supplementary transaction type.
Supplementary Transaction Type Description	Varchar2(60)	Y	N	Description of supplementary transaction type.

140
141

Figure C47: Supplementary Unit Type Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Supplementary Unit Type Code	Number(2)	N	Y	Numeric code indicating the supplementary unit type code.
Supplementary Unit Type Description	Varchar2(60)	Y	N	Description of supplementary unit type code.

142
143

Figure C48: System Log

Attribute	Data Type	Null	Primary Key	Description
System Log ID	Number(12)	N	Y	Unique identification of system Log.
Job Name	Varchar2(40)	N	N	Name of job or process.
Returned Value	Varchar2(25)	Y	N	The returned value, if requested.
Registry Code	Varchar2(3)	Y	N	The registry that job was sent to.
Action Date	DateTime	N	N	Date and time Process was last run
Requested Value	Varchar2(25)	Y	N	The value or parameter requested
Job Status	Varchar2(3)	Y	N	Indication if job is enabled (“E”) or offline (“O”) or failed (“F”)

144
145

Figure C49: System Parameter Entity Details

Attribute	Data Type	Null	Primary Key	Description
Sys Param ID	Number(15)	N	Y	Unique identifier for a system parameter.
System Param Name	Varchar2(65)	Y	N	Name of system parameter.
Param Name1	Varchar2(60)	Y	N	Name of system parameter number one.
Param Value1	Varchar2(60)	Y	N	Value of system parameter number one.
Param Name2	Varchar2(60)	Y	N	Name of system parameter number two.
Param Value2	Varchar2(60)	Y	N	Value of system parameter number two.
Param Name3	Varchar2(60)	Y	N	Name of system parameter number three.
Param Value3	Varchar2(60)	Y	N	Value of system parameter number three.

146
147

Figure C50: Test Status Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Test Status Code	Number(3)	N	Y	Code indicating the status of a test submission.
Test Status Description	Varchar2(60)	Y	N	Text description of a status for a test submission.

148
149

Figure C51: Test Type Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Test Type Code	Number(2)	N	Y	Code indicating a type of test submission.
Test Type Description	Varchar2(60)	Y	N	Text description for the type of test submission.

150
151

Figure C52: Trade Scheme Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Trade Scheme Code	Number(3)	N	Y	Unique identifier of a trading scheme.
Trade Scheme Description	Varchar2(60)	Y	N	Text description of trade scheme.

152
153

Figure C53: Transaction Block Entity Details

Attribute	Data Type	Null	Primary Key	Description
Transaction Block ID	Number(15)	N	Y	Reference to a unit block involved in a transaction.
Original Period Code	Number(2)	Y	N	Code indicating the original Commitment Period of the units in the block.
Transaction ID	Varchar2(20)	N	N	Unique identifier for a transaction.
Start Block	Number(15)	Y	N	Number indicating the beginning of a range of serial numbers.
End Block	Number(15)	Y	N	Number indicating the end of a range of serial numbers.
Applicable Period Code	Number(2)	N	N	Code indicating the currently applicable Commitment Period of the units in the block.
Originating Registry Code	Varchar2(3)	N	N	Code identifying the originating registry.
Project Identifier	Varchar2(12)	Y	N	Country code and Project number which uniquely identify a JI Project.
Block Role	Varchar2(1)	Y	N	Identifier field for unit blocks used for replacement (replaced or to be replaced).
Expiry Date	DateTime	Y	N	Date in which an lCER or tCER expires.
Unit Type Code	Number(2)	N	N	Code indicating the type of units in a block.
Project Freeze Flag	Char(1)	Y	N	Identifies whether an lCER has been frozen.
Supplementary Unit Type Code	Number(2)	N	Y	Numeric code indicating the supplemental unit type code.

(cont.)

154
155

Figure C53: Transaction Block Entity Details (cont.)

Attribute	Data Type	Null	Primary Key	Description
Year in Commitment Period	Number(4)	Y	N	Given year within a Commitment Period. Values are 2005, 2006, 2007, etc.
Acquiring Registry Account	Number(15)	Y	N	Account of Acquiring Registry.
Acquiring Account Type Code	Number(3)	N	N	Numeric code indicating the type of account.
Acquiring Registry Code	Varchar2(3)	N	N	Code identifying the registry.
Transferring Registry Account	Number(15)	Y	N	Account of Transferring Registry.
Transferring Account Type Code	Number(3)	Y	N	Account of Transferring Registry.
Transferring Registry Code	Varchar2(3)	N	N	Code identifying the registry.
Installation ID	Number(15)	Y	N	Identifier of installation number.
Track	Number(2)	Y	N	Code identifying Project track.
LULUCF Code	Number(3)	Y	N	Code identifying the LULUCF activity associated with the units in the block.

Figure C54: Transaction Block History Entity Details

Attribute	Data Type	Null	Primary Key	Description
Transaction Block History ID	Number(15)	N	Y	Identifier for a transaction block history record.
Transaction Block ID	Number(15)	N	N	Reference to a unit block involved in a transaction.
Transaction Block DateTime	DateTime	Y	N	Date and time a transaction block status was updated.
Response Code	Number(5)	N	N	Code indicating a response to a transaction process, such as a transaction validation.

Figure C55: Transaction Log Entity Details

Attribute	Data Type	Null	Primary Key	Description
Transaction ID	Varchar2(20)	N	Y	Unique identifier for a transaction.
Transaction Type Code	Number(2)	N	N	Code indicating the type of transaction.
Transaction Date	DateTime	Y	N	Date on which a transaction was submitted.
Acquiring Registry Code	Varchar2(3)	N	N	Code identifying the Acquiring Registry.
Acquiring Registry Account	Number(15)	Y	N	Account for Acquiring Registry.
Acquiring Account Type	Number(3)	N	N	Code identifying the acquiring account type.
Transferring Registry Account	Number(15)	N	N	Account for Transferring Registry.
Transferring Registry Code	Varchar2(3)	N	N	Code identifying the Transferring Registry.
Transferring Account Type	Number(3)	N	N	Code identifying the transferring account type.
Supplementary Transaction Type Code	Number(2)	Y	N	Code indicating supplementary transaction type.

160
161

Figure C56: Transaction Log History Entity Details

Attribute	Data Type	Null	Primary Key	Description
Transaction Log History ID	Number(15)	N	Y	Unique identifier for a transaction status.
Transaction Status DateTime	DateTime	Y	N	Date and time the transaction status changed.
Transaction ID	Varchar2(20)	N	N	Unique identifier for a transaction.
Transaction Status Code	Number(3)	N	N	Code indicating the transaction status.

162
163

Figure C57: Transaction Status Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Transaction Status Code	Number(3)	N	Y	Code indicating the transaction status.
Transaction Status Description	Varchar2(60)	Y	N	Text description of a transaction status.

164
165

Figure C58: Transaction Type Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Transaction Type Code	Number(2)	N	Y	Code indicating the type of transaction.
Transaction Type Description	Varchar2(60)	Y	N	Text description of a type of transaction.

166
167

Figure C59: Unit Block Entity Details

Attribute	Data Type	Null	Primary Key	Description
Block ID	Number(15)	N	Y	Unique identifier of a unit block.
Start Block	Number(15)	N	N	Number indicating the beginning of a range of serial numbers.
End Block	Number(15)	N	N	Number indicating the end of a range of serial numbers.
Unit Type Code	Number(2)	N	N	Code indicating the type of units in a block.
Original Period Code	Number(2)	N	N	Code indicating the original Commitment Period of the units in the block.
Applicable Period Code	Number(2)	N	N	Code indicating the currently applicable Commitment Period of the units in the block.
Originating Registry Code	Varchar2(3)	N	N	Code identifying the registry.
Holding Registry Code	Varchar2(3)	N	Y	Code identifying the registry currently holding unit block.

(cont.)

168
169

Figure C59: Unit Block Entity Details (cont.)

Attribute	Data Type	Null	Primary Key	Description
Account Type Code	Number(3)	N	N	Numeric code indicating the type of account.
Supplementary Unit Type Code	Number(2)	Y	N	Numeric code indicating the supplemental unit type code.
Unit Status Code	Number(1)	N	N	Code indicating the status of a unit block.
Expiry Date	Date	Y	N	Date in which lCERs or tCERs expire.
Project Freeze Flag	Char(1)	Y	N	Identifies whether an lCER has been frozen.
Last Action Date	DateTime	Y	N	Date and time the last action was performed.
Project ID	Number(12)	N	N	Unique identifier for a Project.
Track	Number(2)	Y	N	Code identifying Project track.
LULUCF Code	Number(3)	Y	N	Code identifying the LULUCF activity associated with the units in the block.
Project Identifier	Varchar2(12)	Y	N	Country code and Project number which uniquely identifies a JI Project.

170
171

Figure C60: Unit Status Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Unit Status Code	Number(1)	N	Y	Code indicating the status of a unit block.
Unit Status Description	Varchar2(60)	Y	N	Text description of a unit block status.

172
173

Figure C61: Unit Type Code Entity Details

Attribute	Data Type	Null	Primary Key	Description
Unit Type Code	Number(2)	N	Y	Code indicating the type of units in a block.
Unit Type Description	Varchar2(60)	Y	N	Test describing the type of units in a block.

174
175

Figure C62: Version History Entity Details

Attribute	Data Type	Null	Primary Key	Description
Version ID	Number(12)	N	Y	Unique identifier for a version.
Application Code	Varchar2(7)	N	Y	Primary key code for application.
Version Comment	Varchar2(3000)	Y	N	Brief explanation of features, corrections or enhancements available in this version.
Version Date	DateTime	Y	N	Date on which a version was implemented.

176
177

Figure C63: Primary Keys

Entity	Key Column	Key Order	Name
Account Type Code	Account Type Code	1	PK Account Type Code
Application	Application Code	1	PK Application
Audit Log	Audit ID	1	PK Audit Log
Check Catalog Code	Check ID	1	PK Check Catalog
Commitment Period Code	Period Code	1	PK Commitment Period
Country Code	Country Code	1	PK Country Code
CreditingPeriod Type Code	Crediting Period Type Code	1	PK Crediting Period Type
Criteria Type Code	Criteria Type Code	1	PK Eligibility
Eligibility Status Code	Eligibility Status Code	1	PK Eligibility Status Code
Inconsistent Block	Inconsistent Block ID	1	PK Inconsistent Block
LULUCF Code	LULUCF Code	1	PK LULUCF Code
Project	Project ID	1	PK Project
Project Action Code	Project Action Code	1	PK Project Action Code
Project Action Log	Project Log ID	1	PK Project Action Log
Project Action Log	Project Action Code	1	PK Project Action Log
Reconciliation Status History	Recon Log ID	1	PK Recon Status History
Registry	Registry Code	1	PK Registry
Registry Attribute	Registry Code	1	PK Registry Attribute

(cont.)

178
179
180

Figure C63: Primary Keys (cont.)

Entity	Key Column	Key Order	Name
Registry Attribute	Attribute Type Code	2	PK Registry Attribute
Registry Eligibility	Reg Eligibility ID	1	PK Registry Eligibility
Registry Notification	Registry Code	1	PK Registry Notification
Registry Notification	Notification ID	1	PK Registry Notification
Registry People	Registry People ID	1	PK Registry People
Registry Status Code	Registry Status Code	1	PK Registry Status Code
Registry Status History	Registry Status ID	1	PK Registry Status History
Registry Test History	Registry Code	1	PK Registry Test History
Registry Trading Scheme	Registry Code	1	PK Registry Trade Scheme
Registry Trading Scheme	Trade Scheme Code	2	PK Registry Trade Scheme
Replacement Unit Block	Block ID	1	PK Relationship Unit Block
Replacement Unit Block	Replacement Block ID	2	PK Relationship Unit Block
Response Catalog	Response Code	1	PK Response Code
Response Log	Response Log ID	1	PK Response Log
Response Scheme Check	Response Code	1	PK Response Scheme Check
Response Scheme Check	Trade Scheme Code	1	PK Response Scheme Check
Route Status Code	Route Status Code	1	PK Route Status Code
Routing Log	Route ID	1	PK Routing Log
Routing Status History	Route History ID	1	PK Route Status History
Supp Transaction Type Code	Supp Transaction Type Code	1	PK Supp Transaction Type Code
Supp Unit Type Code	Supp Unit Type Code	1	PK Supp Unit Type Code
System Log	System Log ID	1	PK System Log
System Parameter	Sys Param ID	1	PK System Parameter
Test Status Code	Test Status Code	1	PK Test Status Code
Test Type Code	Test Type Code	1	PK Test Type Code
Trade Scheme Code	Trade Scheme Code	1	PK Trade Scheme Code
Transaction Block	Transaction Block ID	1	PK Transaction Block
Transaction Block History	Transaction Block History ID	1	PK Transaction Block History

(cont.)

181
182

Figure C63: Primary Keys (cont.)

Entity	Key Column	Key Order	Name
Transaction Log	Transaction ID	1	PK Transaction Log
Transaction Log History	Transaction Log History ID	1	PK Transaction Log History
Transaction Notification	Trans Notification ID	1	PK Transaction Notification
Transaction Status Code	Transaction Status Code	1	PK Transaction Status Code
Transaction Type Code	Transaction Type Code	1	PK Transaction Type Code
Unit Block	Block ID	1	PK Unit Block
Unit Status Code	Unit Status Code	1	PK Unit Status Code
Unit Type Code	Unit Type Code	1	PK Unit Type
Version History	Version ID	1	PK Version History
Version History	Application Code	2	PK Version History

183
184

Figure C64: Foreign Keys

Parent Table	Child Table	Column(s)
Account Type Code	Unit Block	Account Type Code
Account Type Code	Registry Attribute	Account Type Code
Account Type Code	Transaction Log	Transferring Account Type
Account Type Code	Transaction Log	Acquiring Account Type
Account Type Code	Transaction Block	Acquiring Account Type Code
Account Type Code	Transaction Block	Transferring Account Type Code
Application	Version History	Application Code
Attribute Type Code	Registry Attribute	Attribute Type Code
Check Catalog Code	Check Response	Check ID
Commitment Period	Registry Attribute	Period Code
Commitment Period	Transaction Block	Original Period Code
Commitment Period	Transaction Block	Applicable Period Code
Commitment Period	Unit Block	Original Period Code
Commitment Period	Unit Block	Applicable Period Code

(cont.)

Figure C63: Foreign Keys (cont.)

Parent Table	Child Table	Column(s)
Country Code	People	Country Code
Country Code	Project	Country Code
Crediting Period Type Code	Project	Crediting Period Type Code
Criteria Type Code	Registry Eligibility	Criteria Type Code
Eligibility Status Code	Registry Eligibility	Eligibility Status Code
LULUCF Code	Project	LULUCF Code
LULUCF Code	Unit Block	LULUCF Code
Notification	Registry Notification	Notification ID
Notification	Transaction Notification	Notification ID
Notification Method Code	Notification	Notification Method Code
Notification Type Code	Notification	Notification Type Code
People	People Security	People ID
People	Registry People	People ID
People Security	People Security History	People ID
Project	Project Action Log	Project ID
Project	Unit Block	Project ID
Project	Project Action Log	Project ID
Project	Unit Block	Project ID
Project Action Code	Project Action Log	Project Action Code
Project Action Log	Notification	Project Log ID
Project Action Log	Notification	Project Action Code
Project Status Code	Project	Project Status Code
Reconciliation Log	Reconciliation Status History	Recon ID
Reconciliation Log	Message Log	Recon ID
Reconciliation Phase Code	Reconciliation Log	Recon Phase Code
Reconciliation Status Code	Reconciliation Status History	Recon Status Code
Reconciliation Status History	Inconsistent Block	Recon Log ID
Registry	Inconsistent Block	Originating Registry Code
Registry	Message Log	Registry Code

(cont.)

Figure C63: Foreign Keys (cont.)

Parent Table	Child Table	Column(s)
Registry	Reconciliation Log	Registry Code
Registry	Registry Attribute	Registry Code
Registry	Registry Notification	Registry Code
Registry	Registry People	Registry Code
Registry	Registry Status History	Registry Code
Registry	Registry Test History	Registry Code
Registry	Routing Log	Registry Code
Registry	Registry Eligibility	Registry Code
Registry	Transaction Log	Acquiring Registry Code
Registry	Transaction Log	Transferring Registry Code
Registry	Unit Block	Originating Registry Code
Registry	Registry Trading Scheme	Registry Code
Registry	Transaction Block	Registry Code
Registry	Transaction Block	Transferring Registry Code
Registry	Transaction Block	Acquiring Registry Code
Registry	Unit Block	Holding Registry Code
Registry Status Code	Registry Status History	Registry Status Code
Response Catalog	Response Log	Response Code
Response Catalog	Response Scheme Check	Response Code
Response Catalog	Transaction Block History	Response Code
Responsibility Type Code	Registry People	Responsibility Type Code
Route Status Code	Routing Status History	Route Status Code
Routing Log	Routing Status History	Route ID
Supplementary Transaction Type Code	Transaction Log	Supplementary Transaction Type Code
Supplementary Unit Type Code	Unit Block	Supplementary Unit Type Code
Supplementary Unit Type Code	Transaction Block	Supplementary Unit Type Code
Test Status Code	Registry Test History	Test Status Code

(cont.)

Figure C63: Foreign Keys (cont.)

Parent Table	Child Table	Column(s)
Test Type Code	Registry Test History	Test Type Code
Trade Scheme Code	Registry Trading Scheme	Trade Scheme Code
Trade Scheme Code	Response Scheme Check	Trade Scheme Code
Transaction Block	Transaction Block History	Transaction Block ID
Transaction Log	Message Log	Transaction ID
Transaction Log	Routing Log	Transaction ID
Transaction Log	Transaction Block	Transaction ID
Transaction Log	Response Log	Transaction ID
Transaction Log	Transaction Log History	Transaction ID
Transaction Status Code	Transaction Log History	Transaction Status Code
Transaction Type Code	Transaction Log	Transaction Type Code
Unit Block	Replacement Unit Block	Replacement Block ID
Unit Block	Inconsistent Block	Block ID
Unit Status Code	Unit Block	Block ID
Unit Type Code	Unit Block	Unit Type Code
Unit Type Code	Transaction Block	Unit Type Code
Unit Type Code	Project	Unit Type Code

185
186

Annex D
Lookup Tables and Values

This annex defines the codes for all support tables used in this technical specification. These codes define the acceptable values for lookup tables in the ITL database.

Figure D1: List of Lookup Tables

Table	Description
Account Type Code	Identifies the type of unit accounts in a registry.
Attribute Type Code	Attributes describe various traits of a registry. Typical codes identify a registry's reserve, and initial allocation.
Check Catalog Code	Identifies the type of check.
Commitment Period Code	Identifies the current Commitment Period.
Crediting Period Type Code	Identifies crediting period types.
Criteria Type Code	Identifies eligibility types.
Eligibility Status Code	Identifies eligibility status.
LULUCF Activity Code	Identifies Land Use, Land Use Change, and Forestry categories.
Notification Method Code	Identifies methods of how a notification was sent.
Notification Type Code	Identifies notification type codes.
Party Type Code	Defines the role of the Party.
Project Action Code	Identifies action taken by CDM Executive Board.
Project Status Code	Identifies Project status.
Reconciliation Phase Code	Identifies the phase of reconciliation to be initiated.
Reconciliation Status Code	Identifies the status of a reconciliation process.
Registry Status Code	Identifies the operational status of a registry.
Responsibility Type Code	Identifies the type of responsibility a person has to a registry.
Route Status Code	Identifies the status of a message routing.
Test Status Code	Identifies the status of a test.
Test Type Code	Identifies the type of test.
Trade Scheme Code	Lookup table of trade schemes.

(cont.)

Figure D1: List of Lookup Tables (cont.)

Table	Description
Transaction Status Code	Identifies the status of a transaction.
Transaction Type Code	Identifies the type of transaction.
Unit Status Code	Identifies whether the units have been flagged for processing.
Unit Type Code	Identifies the type of unit.

196
197
198
199

Figure D2: Account Type Code

Code	Description
100	Holding Account
110	Pending Account
120	Operator Holding Account
121	Person Holding Account
210	Net Source Cancellation Account (Type 1)
220	Non-compliance Cancellation Account (Type 2)
230	Voluntary Cancellation Account (Type 3)
240	Excess Issuance Cancellation Account (Type 4)
300	Retirement Account
411	tCER Replacement Account for Expiry (Type 1)
421	lCER Replacement Account for Expiry (Type 1)
422	lCER Replacement Account for Reversal in Storage (Type 2)
423	lCER Replacement Account for Non-submission of Certification Report (Type 3)

201

Figure D3: Attribute Type Code

Code	Description
1	Total Allocation AAUs
2	Commitment Period Reserve Limit
3	Annual RMU Issuance
4	RMU Issuance At End of Commitment Period
5	Total Issuable RMUs for Activity Type 1
6	Total Issuable RMUs for Activity Type 2
7	Total Issuable RMUs for Activity Type 3

Figure D4: Check Catalog Code

Code	Description
1	Registry Validation
2	Message Viability
3	Data Integrity
4	Message Sequence for Registry Messages
5	Message Sequence for STL Messages
6	Message Sequence for Reconciliation
7	General Transaction Checks
10	Issuance
20	Conversion
30	External transfer
40	Cancellation
50	Retirement
60	Replacement
70	Carry-over
80	Expiry Date Change
100	Reconciliation Data Integrity

(cont.)

Figure D4: Check Catalog Code (cont.)

Code	Description
110	Reconciliation Message Sequence for Registry Messages
120	Reconciliation Message Sequence for STL Messages
130	Reconciliation Checks
140	Registry Messages

Figure D5: Commitment Period Code

Code	Description
0	Supplementary Program Commitment Period (2005 - 2007)
1	First Commitment Period
2	Second Commitment Period
3	Third Commitment Period
4	Fourth Commitment Period

Figure D6: Crediting Period Type Code

Code	Description
1	30 year
2	20-40-60 year

Figure D7: Criteria Type Code

Code	Description
1	Kyoto Party
2	Assign amount calculations Approved
3	Emissions estimation protocols Approved
4	National registry Approved
5	National inventory report Approved
6	Assign amount adjustments and supplementary information Approved

Figure D8: Eligibility Status Code

Code	Description
1	Approved
2	Approved by default
3	Disapproved
4	Revoked/Suspended
5	Unknown

Figure D9: LULUCF Code

Code	Description
1	Afforestation, reforestation and deforestation activities
2	Forest management activities
3	Cropland management, grazing land management and revegetation
4	Emissions reduction

Figure D10: Notification Method Code

Code	Description
1	Web service
2	Email
3	Telephone
4	Other

Figure D11: Notification Type Code

Code	Description
1	Transaction Expiration (24 hours)
2	Carry-over of Units Required
3	Failure to submit certification report - Initial Notification
4	Failure to submit certification report - Reminder
5	Receipt of certification report - Initial Notification
6	Receipt of certification report - Reminder
7	Reversal in Storage - Initial Notification
8	Reversal in Storage - Reminder
9	Impending Expiration of lCER
10	Impending Expiration of tCER
11	Compliance with Failure to Submit Requirement
12	Compliance with Reversal in Storage Requirement

Figure D12: Party Type Code

Code	Description
1	Initiating Registry
2	Acquiring Registry

Figure D13: Project Action Code

Code	Description
1	Approved
2	Failure to Submit Notification Finding
3	Reversal in Storage Finding
4	Suspended/Revoked
5	Approval of 2nd Credit Period Renewed
6	Approval of 3rd Credit Period Renewed

Figure D14: Project Status Code

Code	Description
1	Pending
2	Approved
3	Unknown
4	Disapproved

Figure D15: Reconciliation Phase Code

Code	Description
1	Totals
2	Unit Block Detail
3	Audit Trail

Figure D16: Reconciliation Status Code

Code	Description
1	"Initiated"
2	"Validated"
3	"ITL Totals Inconsistent"
4	"ITL Unit Blocks Inconsistent"
5	"ITL Completed"
6	"ITL Completed with Manual Intervention"
7	"ITL Start Request Denied"
8	"STL Totals Inconsistent"
9	"STL Unit Blocks Inconsistent"
10	"STL Validated"
11	"STL Completed with Manual Intervention"

Figure D17: Registry Status Code

Code	Description
0	"Full Operation"
1	"Reconciliation Only"
2	"Not Operating"
3	"Temporarily Restricted"
4	"Permanently Restricted"

Figure D18: Responsibility Type Code

Code	Description
1	Registry Manager
2	Database Administrator/System Manager
3	Contact

Figure D19: Route Status Code

Code	Description
1	Sent to STL
2	Received from STL

Figure D20: Test Status Code

Code	Description
1	In Progress
2	Failed
3	Passed
4	Abandoned

Figure D21: Test Type Code

Code	Description
1	Communication Initialisation
2	Website Access
3	Information Retrieval
4	Issuance
5	Conversion
6	Internal Transfer
7	External Transfer
8	Cancellation
9	Retirement
10	Carry-over
11	Replacement
12	Expiry Date Change
13	Reconciliation

Figure D22: Trade Scheme Code

Code	Description
1	European Commission GHG Trading

Figure D23: Transaction Status Code

Code	Description
1	"Proposed"
2	"Checked (No Discrepancy)"
3	"Checked (Discrepancy)"
4	"Completed"
5	"Terminated"
6	"Rejected"
7	"Cancelled"
8	"Accepted"
9	"STL Checked (No Discrepancy)"
10	"STL Checked (Discrepancy)"

Figure D24: Transaction Type Code

Code	Description
1	Issuance - Initial creation of a unit
2	Conversion - Transformation of unit to create an ERU
3	External - External transfer of unit between registries
4	Cancellation - Internal transfer of unit
5	Retirement - Internal transfer of unit
6	Replacement - Replacement of lCER or tCER
7	Carry-over - Extension of unit validity
8	Expiration Date Change
10	Internal - Internal transfer of unit/supplementary program transaction

Figure D25: Unit Status Code

Code	Description
0	Available
1	Not Available Due to Transaction
2	Not Available Due to ITL Inconsistency
3	Not Available Due to Transaction and ITL Inconsistency
4	Not Available Due to STL Inconsistency
5	Not Available Due to STL Inconsistency and Transaction
6	Not Available Due to STL Inconsistency and ITL Inconsistency
7	Not Available Due to STL Inconsistency, Transaction, and ITL Inconsistency

Figure D26: Unit Type Code

Code	Description
0	Non-Kyoto Unit
1	AAU - Assigned Amount Unit
2	RMU - Removal Unit
3	ERU - Emission Reduction Unit
4	ERU - Converted from an RMU
5	CER - Certified Emission Reduction Unit converted from an AAU
6	tCER - Temporary CER
7	lCER - Long Term CER

Annex E
Functions and Web services

Table of Functions

Annex E
Functions and Web services

1. Introduction

The following sections explain the conventions and structured data storage utilised in the specification for each function.

1.1 Result Identifier

Each function returns a Result Identifier. This value will indicate whether the function succeeded or failed. A failure could be the result of a business decision (a failed check) or the result of an unanticipated exception error (a run time error). The convention used here is that zero (0) indicates failure and one (1) indicates success.

1.2 Transaction Object

The Transaction Object is used to store all the elements involved in a transaction required for processing. The Transaction Object is a parent class to the TransactionUnitBlock Object. The structure of the Transaction Object is as follows:

Structure TransactionObject
- TransactionIdentifier As String
- TransactionType As Integer
- SuppTransactionType As Integer
- TransactionStatus As Integer
- TransactionStatusDate As DateTime
- TransferringRegistryIdentifier As String
- TransferringRegistryAccountType As Integer
- TransferringRegistryAccountIdentifier As Long
- AcquiringRegistryIdentifier As String
- AcquiringRegistryAccountType As Integer
- AcquiringRegistryAccountIdentifier As Long
- NotificationIdentifier As Integer
- TransactionBlocks (array) As UnitBlock Object

1.3 UnitBlock Object

The functions below reference a UnitBlockObject to obtain the unit blocks associated with a transaction. An array of UnitBlockObjects is often called for because more than one block can be associated with a transaction. This UnitBlockObject is used in two ways. It is referenced as ITLUnitBlockObject when it holds data retrieved from the ITL Unit Block table. It is referenced as RegistryUnitBlock when it holds data submitted from a Registry through either a Proposal or notification. When the object is used for data from the ITL, the Block ID field will reference the unique identifier of the block's record in the Unit Block table. Data submitted by a registry will not contain Block ID. The structure of the UnitBlockObject is as follows:

Structure UnitBlockObject
- UnitSerialBlockStart As Long
- UnitSerialBlockEnd As Long
- OriginatingRegistryIdentifier As String
- UnitType As Integer
- SuppUnitType As Integer
- OriginalCommitPeriod As Integer
- ApplicableCommitPeriod As Integer
- LULUCFActivity As Integer
- ProjectIdentifier As Integer
- Track As Integer
- BlockRole As String
- TransferringRegistryAccountType As Integer
- TransferringRegistryAccountIdentifier As Long
- TransferringRegistryIdentifier As String
- AcquiringRegistryAccountType As Integer
- AcquiringRegistryAccountIdentifier As Long
- AcquiringRegistryIdentifier As String
- YearInCommitmentPeriod As Integer
- InstallationIdentifier As Long
- ExpiryDate As DateTime
- BlockID As Number
- UnitStatusCode As Integer

1.4 MessageUnitBlock Object

The MessageUnitBlock Object is used to return only the attributes that uniquely identify it and is generally returned as part of a notification to a Registry in conjunction with a CheckResponse Object.

Structure MessageUnitBlockObject
- UnitSerialBlockStart As Long
- UnitSerialBlockEnd As Long
- Originating Registry Identifier As String

1.5 CheckResponse Object

The CheckResponse Object is used to collect all checks that are applicable to a process. As each check is evaluated, the result identifier is modified to indicate a success or failure for the check. When the CheckResponse Object is returned, it only includes those response codes that have failed.

Structure CheckResponseObject
- Result Identifier As Integer
- Response Code As Integer

1.6 EvaluationResult Object

This object stores the specific CheckResponse Object along with the affected Unit Blocks.

Structure EvaluationResultObject
    CheckResponseObject
    UnitBlock (array) As RegistryUnitBlockObject

1.7 Reconciliation Object

The Reconciliation Object is used to describe a reconciliation action and its current status.

Structure ReconciliationObject
    ReconciliationIdentifier As String
    ReconciliationBeginDate As DateTime
    ReconciliationStatusCode As Integer
    ReconciliationEndDate As DateTime
    ReconciliationSnapshotDateTime As DateTime
    ReconciliationStatusDateTime As DateTime
    ReconciliationPhaseCode As Integer

1.8 Totals Object

The Totals Object is used by the reconciliation process to store the number of units held by a registry, account type, or account.

Structure TotalsObject
    ReconciliationIdentifer As String
    ReconciliationSnapshotDateTime As DateTime
    HoldingRegistry As String
    AccountType As Integer
    AccountIdentifier As Integer
    UnitType As Integer
    SupplementaryUnitType As Integer
    UnitCount As Integer

2. Specified Functions

The following functions will be implemented to carry out the duties of the ITL. They are listed in alphabetical order.

512 **Figure E1: AcceptNotification (Web service)**
513

Purpose
This is the HTTP SOAP request that registries use to send notifications regarding transactions in process. Once the request has passed preliminary checks and been written to the queue, an HTTP SOAP response of success or an HTTP SOAP fault of failure is returned. This Web service is hosted on both the ITL and the Registry.
Inputs
TransactionObject, RegistryUnitBlockObject, CheckResponseObject
Table(s)
This function does not interact with the database.
Process
This Web service calls the preliminary checks before writing message to the message queue.
Outputs
Result_Identifier
Call(s)
Preliminary_Checks

514

Figure E2: AcceptProposal (Web service)

Purpose
This is the HTTP SOAP request that registries use to send proposals. Once the request has passed preliminary checks and is written to the message queue, a HTTP SOAP response of success is returned. This Web service is hosted on both the ITL and the Registry.
Inputs
TransactionObject, RegistryUnitBlockObject
Table(s)
This function does not interact with the database.
Process
This Web service calls the preliminary checks before writing message to message queue.
Outputs
Result_Identifier
Call(s)
Preliminary_Checks

518 **Figure E3: CER_Expired_Check (Function)**
519

Purpose
Identify tCERs and lCERs that are to expire in 30 days and notify the registries that hold these units.
Inputs
Table(s)
Process
Every 24 hours, search for tCERs and lCERs that will expire within 30 days. Select Holding_Registry_Code, Originating_Registry_Code, Start_Block, End_Block From Unit_Block Where Unit_Type = (6,7) and Expiration_Date – Current Date < 30 days If any records found, insert record into the Notification table for Impending Expiration 9 (for lCERs) or 10 (for tCERs) For each Holding_Registry_Code represented in the above result set, Insert record into Registry_Notification table. Set Response Code to 6002 – The following tCERs and lCERs units will expire within 30 days. Units that have expired must either be cancelled or replaced. Compose warning message. The message begins with the following text: Notification ID #: The following tCERs and lCERs will expire within 30 days. Units that have expired must either be cancelled or replaced. Write identifying information to the message for each unit block. Write the Originating Registry, Start Block, and End Block to the message. Call AcceptMessage with warning message on the Holding Registry.

(cont.)

Figure E3: CER_Expired_Check (Function)

Outputs
Result_Identifier
Call(s)

521 **Figure E4: Check_Reconciliation_Message (Function)**
522

Purpose
The ITL will execute this function every time a reconciliation message is retrieved from the message queue. This function will in turn execute the checks for message age, reconciliation data integrity checks, and the reconciliation message sequence checks. If any of the checks fail, this message will send a notification of failure to the registry from which the message was received. If all the checks pass, the message will be routed to the next phase as appropriate.
Inputs
ReconciliationObject, TransactionObject
Table(s)
Process
Begin database transaction Call Retrieve_From_Message_Queue. Set Select for update database transaction. Call Write_To_Message_Log. Call Check_Registry_Status. If this function fails, stop processing message and call Reconciliation_Failure_Notification. Call Message_Age_Check. If the function fails, stop processing message and call Reconciliation_Failure_Notification. Call Reconciliation_Data_Integrity_Check, followed by Reconciliation_Message_Sequence_Check. If any check in either function fails, call Reconciliation_Failure_Notification after both functions have completed (so more than one error may be collected). Call Remove_From_Message_Queue so the message will not be processed again. Commit the database transaction. Determine message sender. If it was an STL, call Evaluate_STL_Reconciliation_Message. If it was any other registry, call Evaluate_Registry_Reconciliation_Message.
Outputs
ResultIdentifier ReconciliationObject TransactionObject

(cont.)

Figure E4: Check_Reconciliation_Message (Function) (cont.)

Call(s)
Retrieve_From_Message_Queue Message_Age_Check Write_To_Message_Log Check_Registry_Status Reconciliation_Data_Integrity_Check Reconciliation_Message_Sequence_Check Evaluate_Registry_Reconciliation_Message Evaluate_STL_Reconciliation_Message Reconciliation_Failure_Notification Remove_From_Message_Queue

523

524 **Figure E5: Check_Registry_Status (Function)**
525

Purpose
This function performs a lookup and evaluation of the status of a registry to determine if it is allowed to participate in a transaction or reconciliation action.
Inputs
ReconciliationObject, TransactionObject
Table(s)
Registry, Registry_Status_History
Process
Evaluate the status of the Initiating Registry by retrieving and executing status checks. Call Get_Checks to return the registry status checks. Call Execute_Checks to perform the check. If any check fails, return failure.
Outputs
CheckResponseObject
Call(s)

526

527 **Figure E6: Check_Version (Function)**
528

Purpose
This function compares the major and minor version number in the HTTP SOAP request. These values are checked against the current version of the ITL. Major version numbers must match or the request will be rejected. Messages with an old minor version are still processed, although a warning is issued.
Inputs
Major Version, Minor Version
Table(s)
This function does not interact with the database.
Process
Compare Major Version Number If not a match, return message 1031. Return HTTP SOAP response of failure. If a match, then compare Minor Version Number. If not a match, return message 1032.
Outputs
CheckResponseObject Result_Identifier
Call(s)

529

Figure E7: Close_Reconciliation_Action (Function)

Purpose
This function will update the Reconciliation_Log table with the end date of a reconciliation action.
Inputs
ReconciliationObject
Table(s)
Reconciliation_Log
Process
Update Reconciliation_Log table so that Recon_End_Date = input reconciliation end date.
Outputs
ResultIdentifier
Call(s)

Figure E8: Data_Integrity_Check (Function)

Purpose
This function will check the data in the TransactionObject to ensure that it meets the minimum requirements to begin processing the proposed transaction. See Section 5.4.5 for a list of all data integrity checks performed by this function.
Inputs
TransactionObject, RegistryUnitBlockObject, ITLUnitBlockObject
Table(s)
Check_Catalog, Response_Catalog
Process
For each check identified in Section 5.4.5, evaluate the TransactionObject against the check. Each check will compare data from the registry with data from the ITL. For each check, collect the appropriate response code returned and store in the CheckResponseObject. If there is a failure of a data integrity check, exit this function. If there is no failure, call the Message_Sequence_Check function to continue processing the transaction.
Outputs
CheckResponseObject TransactionObject
Call(s)
Message_Sequence_Check

536 **Figure E9: Delete_Inconsistent_Block (Function)**
537

Purpose
This function will delete records in the Inconsistent Block table as part of clearing blocks from a reconciliation process. It will also update the unit status in the Unit_Block table.
Inputs
ReconciliationObject
Table(s)
Inconsistent_Block, Unit_Block
Process
For each unit block in the UnitBlock array of the ReconciliationObject, delete record in the Inconsistent_Block table Update unit_status in Unit_Block table If the unit_status = 2,3,6,7 unit_status = unit_status – 2 Update Last_Action_Date in Unit_Block table with the current DateTime.
Outputs
Result_Identifier Inconsistent_Block_ID
Call(s)

Figure E10: Delineate_Units (Function)

Purpose
This function updates the status of a unit block so that the units are either available for transactions or are unavailable due to some ongoing transaction or reconciliation process. If the unit status is greater than zero, the unit cannot be traded. The unit status also keeps track of the reason the unit cannot be traded through a binary code system. When the unit is part of an ongoing transaction a value of 1 is added to the current unit status. When unit is part of an inconsistent block identified by ITL reconciliation a value of 2 is added to the current unit status. When the unit is part of an inconsistent block identified by STL reconciliation, a value of 4 is added to the current unit status. When the value of unit status is 1, 3, 5, or 7 the unit is part of an ongoing transaction. When the value is 2, 3, 6, or 7 the unit is part of an inconsistent block identified by ITL reconciliation. When the value is 4, 5, 6, or 7 the unit is part of an inconsistent block identified by STL reconciliation. The unit status should reflect only one instance of a unit status, even if multiple processes have tied up the unit for the same reason.
Inputs
TransactionObject, ITLUnitBlockObject, UnitStatusCode, Engaged_flg
Table(s)
Unit_Block
Process
For each transaction block in the TransactionObject and associated ITLUnitBlockObjects, update the Unit_Status_Code in the Unit_Block table as described below. If the parameter “engaged_flg” is passed as true, and the existing unit status code is not equal to 1, then add 1 to the unit status code value. If parameter “engaged_flg” is passed as false, then subtract 1 from the unit status code value. Update the Unit_Status_DateTime fields with the system DateTime. For each transaction block in the TransactionObject and associated ITLUnitBlockObjects, update the unit_status in the Unit_Block table as described below.

(cont.)

Figure E10: Delineate_Units (Function) (cont.)

Process
When engaged_flg = 1 (ongoing transaction), If the unit_status <> 1,3,5,7 Unit_status = Unit_status + 1 When with engaged_flg = 0 (free to be traded), If the unit_status = 1,3,5,6 unit_status = unit_status -1
Outputs
Call(s)

542

543 **Figure E11: Determine_Route_For_Proposal (Function)**
544

Purpose
After a proposed transaction has been checked by the ITL, this function will route the transaction to the next appropriate party based on the transaction status and transaction type.
Inputs
TransactionObject
Table(s)
This function does not interact with the database.
Process
Evaluate the Transaction Status If the Transaction Status = 3 ("Checked (Discrepancy)"), Call Send_Notification_To_Registry with Transferring Registry code to inform the Transferring Registry of the discrepancy. If the transaction type = 3 (External) Call Send_Notification_To_Registry with the Acquiring Registry code. If the Transaction Status = 2 ("Checked (No Discrepancy)"), Check to see if either party to the transaction is in a supplementary program. Call Trade_Scheme_Member for the Transferring Registry, and, if this is an External Transaction (3), the Acquiring Registry. If either function call returns success, call Send_To_STL. If the parties are not in a supplementary program, route the message as appropriate for the transaction type. For type 3 (External Transfers), call Send_Proposal_To_Registry for the Acquiring Registry. For all other types of transactions, call Send_Notification_To_Registry for the Transferring Registry.
Outputs
ResultIdentifier
Call(s)
Send_Notification_To_Registry Trade_Scheme_Member Send_To_STL Send_Proposal_To_Registry

545 **Figure E12: Evaluate_Registry_Reconciliation_Message (Function)**
546

Purpose
This function checks the reconciliation status of an incoming message from a registry and routes the message as appropriate.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
Begin database transaction. Evaluate the reconciliation status: If "Initiated" (1), Call Validate_Totals If "Validated" (2), Call Forward_Totals_To_STL If "Totals Inconsistent" (3), Call Validate_Unit_Blocks If "Unit Blocks Inconsistent" (4) Call Accept_Audit_Trail If "STL Totals Inconsistent" (8), Call Forward_Unit_Blocks_To_STL If "STL Unit Blocks Inconsistent" (9), Call Forward_Audit_Trail_To_STL. When the process returns from one of the above functions, call Remove_From_Message_Queue and commit the database transaction.
Outputs
CheckResponseObject

(cont.)

547

Figure E12: Evaluate_Registry_Reconciliation_Message (Function) (cont.)

Call(s)
Validate_Totals Forward_Totals_To_STL Validate_Unit_Blocks Forward_Unit_Blocks_To_STL Forward_Audit_Trail_To_STL Accept_Audit_Trail Remove_From_Message_Queue

548 **Figure E13: Evaluate_ResponseObject (Function)**
549

Purpose
This function looks at the CheckResponseObject and determines if the transaction failed to meet the minimum checks required for further processing.
Inputs
CheckResponseObject
Table(s)
Response_Code
Process
Check the response codes collected in the CheckResponseObject. If any codes indicate a failure (Response_Code.Response_Type_Code = 0), then return HTTP SOAP response of failure and include response codes in CheckResponseObject indicating nature of failure. If there are no failures in the CheckResponseObject, then call the Evaluate_Transaction function to determine the next level of checks to perform.
Outputs
CheckResponseObject
Call(s)
Evaluate_Transaction

550

551 **Figure E14: Evaluate_STL_Reconciliation_Notice (Function)**
552

Purpose
This function checks the reconciliation status on incoming messages from the STL and processes them or routes them as appropriate.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
Begin database transaction. Update the reconciliation status on the ITL with the status passed from the STL. Then evaluate the reconciliation status: If Initiated (1) Call Start_Reconciliation If "STL Totals Inconsistent" (8) Call Write_To_Reconciliation_Status Call Request_Unit_Blocks_From_Registry If "STL Unit Blocks Inconsistent" (9) Call Insert_Inconsistent_Block for each block Call Write_To_Reconciliation_Status Call Request_Audit_Trail_From_Registry If "STL Validated" (10) Call Write_To_Reconciliation_Status Call Send_Reconciliation_Notification_To_Registry Call Delete_Inconsistent_Block to free any blocks previously frozen If "STL Complete with Manual Intervention" (11) Call Write_To_Reconciliation_Status Call Send_Reconciliation_Notification_To_Registry. When the process returns from one of the above functions, commit the database transaction.

(cont.)

Figure E14: Evaluate_STL_Reconciliation_Notice (Function) (cont.)

Outputs
CheckResponseObject
Call(s)
Start_Reconciliation Request_Unit_Blocks_From_Registry Request_Audit_Trail_From_Registry Send_Reconciliation_Notification_To_Registry Delete_Inconsistent_Block Start_Reconciliation Write_To_Reconciliation_Status Close_Reconciliation_Action

Figure E15: Evaluate_Transaction (Function)

Purpose
This function determines what the next action should be in processing the transaction. This is accomplished by checking both the transaction status and the transaction type.
Inputs
TransactionObject
Table(s)
Process
Evaluate the transaction status. If the status = "Proposed", then evaluate the transaction type. All transaction types are processed by the ITL (Validate_Proposal). After control is passed back to this function, determine where the TransactionObject is to be routed next (Determine_Route_For_Proposal). If the transaction is anything other than "Proposed", then the transaction is "in process." If the transaction type = 3 (External) then initiate external processing (Process_External_Notification), else, all other transaction types (1, 2, 4, 5, 6, 7, 8 and 10) are processed (Process_Non-external_Notification).
Outputs
TransactionObject
Call(s)
Validate_Proposal Process_Non-external_Notification Process_External_Notification Determine_Route_For_Proposal

Figure E16: Execute_Checks (Function)

Purpose
This function will execute each check contained in the CheckResponseObject. It will update the result of each check in the ResponseObject.
Inputs
CheckResponseObject, TransactionObject, RegistryUnitBlockObject, ITLUnitBlockObject
Table(s)
Process
For each check in the CheckResponseObject, perform check against the data in the RegistryUnitBlockObject. If a check returns a failure, record the failure of that check in the CheckResponseObject and call the Write_Block_History function.
Outputs
CheckResponseObject
Call(s)
Write_Block_History

559 **Figure E17: Finalise_Transaction (Function)**
560

Purpose
When a transaction is complete, this function is called to update the unit holding records of the ITL.
Inputs
TransactionObject
Table(s)
This function does not interact with the database.
Process
This function evaluates the transaction type in order to know how to update the Unit_Block table. If the transaction type = 1 (Issuance) or 10 and the supplementary transaction type = 51 or 54, the Insert_Unit_Block function is then called for every unit block involved in the transaction to add a new record to the Unit_Block table. If the transaction type = 2, 7, or 8 (Conversion, Carry-over, or Expiration Date Change), Delineate_Units is called with 0 for every block in the transaction to indicate that the blocks are free to be used in a trade. The Update_Unit_Block function is then called to change the unit type (for Conversion), the applicable commitment period (for Carry-over), or the expiration date (for Expiration Date Change). If the transaction type = 3, 4, 5 (External, Cancellation, Retirement), or 10 and the supplementary transaction type = 53, 21, 55, 01, 02, 41 Delineate_Units is called with 0 for every block in a transaction to indicate that the blocks are free to be used in a trade. The Update_Block_Ownership_Or_Account_Type function is then called to transfer ownership of the units to the acquiring party. If the transaction type = 6 (Replacement), Delineate_Units is called with 0 for every block in the transaction to indicate that the blocks are free to be used in a trade. Next call Update_Block_Ownership_Or_Account_Type to complete the transfer to the replacement account. Finally, call Replace_Unit_Block to insert a record in the Replacement_Unit_Block table and establish the relationship between the two types of blocks. If the transaction type = 10 and supplementary transaction type = [to be determined], Delineate_Units is called with 0 for every block in the transaction to check that the blocks are free to be used in a trade. Next call Update_Block_Ownership_Account_Type to update the account type and the Supplementary Unit Type code for non-Kyoto Transactions.

(cont.)

Figure E17: Finalise_Transaction (Function) (cont.)

Outputs
Result Identifier
Call(s)
Delineate_Units Insert_Unit_Block Update_Unit_Block Update_Block_Ownership_Or_Account_Type Replace_Unit_Block

561

562 **Figure E18: Forward_Audit_Trail_To_STL (Function)**
563

Purpose
This function writes to the message log and then calls the ReceiveAuditTrail Web service on the STL.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
Call Write_To_Message_Log Call ReceiveAuditTrail Web service on the STL.
Outputs
CheckResponseObject
Call(s)
Write_To_Message_Log

564

565 **Figure E19: Forward_Totals_To_STL (Function)**
566

Purpose
This function writes to the message log and then calls the ReceiveTotals Web service on the STL.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
Call Write_To_Message_Log Call ReceiveTotals Web service on the STL.
Outputs
CheckResponseObject
Call(s)
Write_To_Message_Log

567

568 **Figure E20: Forward_Unit_Blocks_To_STL (Function)**
569

Purpose
This function writes to the message log and then calls the ReceiveUnitBlocks Web service on the STL.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
Call Write_To_Message_Log Call ReceiveUnitBlocks Web service on the STL.
Outputs
CheckResponseObject
Call(s)
Write_To_Message_Log

570

571 **Figure E21: Get_Block_ID (Function)**
572

Purpose
This function will search the unit block table for the serial numbers passed. If the entire range of serial numbers are found, this function returns true and the block ID(s) in which the range of numbers was found is also returned. If all the serial numbers are not found, this function returns false. The block ID(s) of any overlapping blocks are still returned. This is used by the reconciliation process for identifying inconsistent blocks.
Inputs
Unit Block ID
Table(s)
Unit_Block
Process

Step 1: Search for exact match

```
Select Block_ID
From Unit_Block
Where Originating_Registry_Code = input originating registry code
        and Start_Block = input start block
        and End_Block = input end block

If found
        add Block_ID to output Block ID's
        set result to True
else
        proceed to step 2.
```

Step 2: Search for single block that complely encompasses input block

```
Select Block_ID
From Unit_Block
Where Originating_Registry_Code = input originating registry code
        and Start_Block <= input start block
        and End_Block >= input end block

If found
        add Block_ID to output Block ID's
        set result to True
else
        proceed to step 3.
```

(cont.)

Figure E21: Get_Block_ID (Function) (cont.)

Process
Step 3: Search for all unit blocks that overlap with the input serial numbers Search Unit_Block for any block containing either the starting number or ending number. Order by serial number Select Block_ID From Unit_Block Where Originating_Registry_Code = input originating registry code And ((Start_Block >= input start block and Start_Block <= input end block) or (End_Block <= input start block and End_Block <= input end block)) Order By Start_Block Check to see if the blocks returned are contiguous Initialize v_prev_end_block to Start_Block of first record returned + 1 For Every Record Found, Add block id to ouput block ids If Start_Block <> v_prev_end_block + 1, the blocks are not contiguous set result to False and continue searching set v_prev_end_block = End_Block Skip to next record If the records returned are all contiguous, Return True else Return False
Outputs
Result Identifier, array of Block IDs
Call(s)
None

573

574 **Figure E22: Get_Checks (Function)**
575

Purpose
This function returns all the general checks and all of the checks associated with the transaction type which are associated with a check category. The checks are returned in the CheckResponseObject.
Inputs
TransactionObject, Check_Category_Code
Table(s)
Response_Catalog
Process
Retrieve from the Response_Catalog table all general checks and transaction-specific checks where Check_Catalog_Code matches input Check_Category_Code. Store checks to perform in CheckResponseObject.
Outputs
CheckResponseObject
Call(s)

576

577 **Figure E23: InitiateReconciliation (Web service)**
578

Purpose
This is the HTTP SOAP request that an STL will use to to request the ITL to begin a reconciliation action for a specified registry and snapshot DateTime. The ITL will also use this request to inform the STL that the ITL has initiated a reconciliation for a specified registry and snapshot time on its own accord. This will allow the STL to create its own snapshot at the appropriate time. Once the request has passed preliminary checks and been written to the queue, an HTTP SOAP response of success or an HTTP SOAP fault of failure is returned. This Web service is hosted on both the ITL and the STL.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
This Web service calls the preliminary checks before writing message to the message queue.
Outputs
Result_Identifier
Call(s)
Preliminary_Checks Write_To_Message_Queue

579

Figure E24: Insert_Inconsistent_Block (Function)

Purpose
The ITL will call this function to insert a unit block to the Inconsistent_Block table for the registry. The Unit_Status and Unit_Status_DateTime will also be updated in the Unit_Block table. If the unit status is greater than zero, the unit cannot be traded. The unit status also keeps track of the reason the unit cannot be traded through a binary code system. When the unit is part of an ongoing transaction a value of 1 is added to the current unit status. When unit is part of an inconsistent block identified by ITL reconciliation a value of 2 is added to the current unit status. When the unit is part of an inconsistent block identified by STL reconciliation, a value of 4 is added to the current unit status. So, when the value of unit status is 1, 3, 5, or 7 the unit is part of an ongoing transaction. When the value is 2, 3, 6, or 7 the unit is part of an inconsistent block identified by ITL reconciliation. When the value is 4, 5, 6,or 7 the unit is part of an inconsistent block identified by STL reconciliation. The unit status should reflect only one instance of a unit status, even if multiple processes have tied up the unit for the same reason.
Inputs
ReconciliationObject
Table(s)
Process
For each unit block in the UnitBlock array of the ReconciliationObject, insert a record in the Inconsistent_Block table Update unit_status in Unit_Block table If the unit_status <> 2, 3, 6, 7 unit_status = unit_status + 2 Update Last_Action_Date in Unit_Block table with current DateTime.

(cont.)

Figure E24: Insert_Inconsistent_Block (Function) (cont.)

Outputs
ResultIdentifier
Call(s)

582 **Figure E25: Lack_of_Certification_Report (Function)**
583

Purpose
This function will set the freeze flag for all units associated with a project that has failed to submit a certification report. Once the freeze flag is set the units can no longer be traded. This function will also inform all affected registries about the unit freeze.
Inputs
Project ID
Table(s)
Process
Set freeze flag for all units associated with the specified project. Freeze all lCERs for that project Update Unit_Block Set Project_Feeze_Flag = 1 Where Project_ID = input Project ID Insert record into the Notification table for "Failure to Submit Certification Report" (type 3). For each Holding_Registry_Code represented in the above result set, Insert record into Registry_Notification table. Set Response Code to 6002 – lCERs may not be traded because a certification report was not submitted for the Project. Compose warning message. The message begins with the following text: Notification ID #: The following lCERs may not be traded because a certification report was not submitted for the Project. Write identifying information to the message for each unit block. Write the Originating Registry, Start Block, and End Block to the message. Call AcceptMessage with warning message on Holding Registry.

(cont.)

Figure E25: Lack_of_Certification_Report (Function) (cont.)

Outputs
Result_Identifier
Call(s)

584

585 **Figure E26: Message_Age_Check (Function)**
586

Purpose
This function will evaluate whether the message retrieved from the queue was received more than 24 hours earlier. If so, a response code is returned and the processing is terminated.
Inputs
Message DateTime
Table(s)
No interaction with the database
Process
If current time – Message DateTime > 24 hours, return response code 1301
Outputs
Result_Identifier
Call(s)

587

588 **Figure E27: Message_Sequence_Check (Function)**
589

Purpose
This function will check the data in the TransactionObject to ensure that the transaction status identified in the TransactionObject is in the expected sequence order. See Sections 5.7 and 5.8 for specifics on the sequence checks.
Inputs
TransactionObject
Table(s)
Response_Catalog, Transaction_Log, Transaction_Log_History
Process
For each check identified in Sections 5.7 and 5.8 (depending on whether the message is from a registry or an STL), evaluate the TransactionObject against the check. Update ResultIdentifier and ResponseCode in the CheckResponseObject with the result of the check.
Outputs
CheckResponseObject TransactionObject
Call(s)

590

591 **Figure E28: Outstanding_Unit_Cleanup (Function)**
592

Purpose
Identify units still in holding accounts for a Commitment Period that has already expired. Units should be cancelled, retired, carried-over or used for replacement.
Inputs
Commitment Period ID
Table(s)
Process
Search the Unit_Block table for registries holding units from the specified Commitment Period. Select Holding_Registry_Code, Originating_Registry_Code, Start_Block, End_Block From Unit_Block Where Applicable_Period_Code = input Commitment Period And Account_Type_Code = 100, 110, 120, or 121 If any records found, insert record into to Notification table for "Carry-over of Units Required" (type 2). For each Holding_Registry_Code represented in the above result set, Insert record into Registry_Notification table. Set Response Code to 6001 – Units that are not applicable to the current or a future Commitment Period must be cancelled or carried-over. Compose warning message. The message begins with the following text: Notification ID#: The following units will no longer be applicable to the current Commitment Period. Please take action to carry-over or cancel these units. Write to the message the identifying information for each unit block. Write the Originating Registry, Start Block, and End Block to the message. Call AcceptMessage with warning message on the Holding Registry.

(cont.)

Figure E28: Outstanding_Unit_Cleanup (Function) (cont.)

Outputs
ResultIdentifier
Call(s)

593

Figure E29: Preliminary_Checks (Function)

Purpose
The ITL calls this function when a message is received through one of its Web services. This function will authenticate the sender of the message and check the version number. If the message contains a transaction, it will call another function to write the message to a file. If there are no problems, the message will be added to the message queue.
Inputs
TransactionObject, ReconciliationObject
Table(s)
This function does not interact with the database.
Process
Call Registry_Authentication. Call Check_Version. If the version check returns failure, this function immediately returns failure. If this message contains a transaction, call Write_To_File to record the contents of the message. If there is no problem in the above process, call the Write_To_Message_Queue function.
Outputs
Result_Identifier
Call(s)
Check_Version Write_To_File Write_To_Message_Queue

597 **Figure E30: Process_External_Notification (Function)**
598

Purpose
This function processes a transaction after an update has been received regarding an External Transaction (transaction type 3) or 10 and Supplementary Transaction Type 21. This function will update the status at the ITL, and route the transaction to the next party based on the transaction status.
Inputs
TransactionObject
Table(s)
Process
The first step of this function is to update the transaction status at the ITL by calling the Write_Transaction_Status function. Once complete, this function evaluates the transaction status. If the Transaction Status = 4 ("Complete"), the transaction was completed at the Transferring Registry and the ITL calls Finalise_Transaction to update its own records. If either party to the transaction, as determined by calling Trade_Scheme_Member, is in a supplementary program, Send_To_STL is also called. If the Transaction Status = 5 ("Terminated"), this function simply calls Delineate_Units with 0 to indicate that the units involved in the transaction are free to be used in another transaction. In the same manner as above, Send_To_STL is also called if either party to the transaction is in a supplementary program. If the Transaction Status = 6 ("Rejected"), the Acquiring Registry refused the transaction. Send_Notification_To_Registry for the Transferring Registry is called to inform that party of the rejection. If either party is in in a supplementary program, Send_To_STL is also called. If the Transaction Status = 8 ("Accepted"), the Acquiring Registry approved the transaction. Call Send_Notification_To_Registry to inform the Transferring Registry of the Acquiring Registry's decision.

(cont.)

599

Figure E30: Process_External_Notification (Function) (cont.)

Outputs
CheckResponseObject
Call(s)
Write_Transaction_Status Send_Notification_To_Registry Trade_Scheme_Member Send_To_STL Finalise_Transaction Delineate_Units

600 **Figure E31: Process_Non-external_Notification (Function)**
601

Purpose
After the ITL receives an update regarding any transaction except an External Transaction (transaction type 3), this function will update the status at the ITL, and route the transaction to the next party based on the transaction status.
Inputs
TransactionObject
Table(s)
Process
The first step of this function is to update the transaction status at the ITL by calling the Write_Transaction_Status function. Once complete, this function evaluates the transaction status. If the Transaction Status = 4 ("Complete"), the transaction was completed at the Transferring Registry, and the ITL calls Finalise_Transaction to update its own records. If the transferring registry, as determined by calling Trade_Scheme_Member, is in a supplemental program, Send_To_STL is also called. If the Transaction Status = 5 ("Terminated"), this function simply calls Delineate_Units with 0 to indicate that the units involved in the transaction are free to be used in another transaction. In the same manner as above, Send_To_STL is also called if the Transferring Registry is in a supplemental program.
Outputs
CheckResponseObject
Call(s)
Trade_Scheme_Member Send_To_STL Finalise_Transaction Delineate_Units Replace_Unit_Block

602

603 **Figure E32: ProvideAuditTrail (Web service)**
604

Purpose
This is the HTTP SOAP request that the STL uses to initiate the request for the ITL to request audit trail data from a registry. If the SOAP request is not accepted by the ITL, a Registry HTTP SOAP fault is returned. Once the request has passed reconciliation message checks, an HTTP SOAP response is returned. The ITL will relay the request to the appropriate registry. This Web service is hosted by the ITL and the Registry.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
This Web service calls the Preliminary_Checks function to validate the incoming message and add the message to the queue.
Outputs
CheckResponseObject
Call(s)
Preliminary_Checks

605

606 **Figure E33: ProvideTotals (Web service)**
607

Purpose
This is the HTTP SOAP request that the STL uses to initiate a request for total number of units held by each account at a registry. If the SOAP request is not accepted, an HTTP SOAP fault is returned. Once the request has passed reconciliation message checks and been written to the queue, an HTTP SOAP response is returned. The ITL relays the request to the appropriate registry. This Web service is hosted on the ITL and the Registry.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
This Web service calls the Preliminary_Checks function to validate the incoming message and add the message to the queue.
Outputs
CheckResponseObject
Call(s)
Preliminary_Checks

608

609 **Figure E34: ProvideUnitBlocks (Web service)**
610

Purpose
This is the HTTP SOAP request that the STL uses to initiate a request for the unit blocks held by an account at a registry. If the SOAP request is not accepted, an HTTP SOAP fault is returned. Once the request has passed reconciliation message checks and been written to the queue, an HTTP SOAP response is returned. The ITL relays the request to the appropriate registry. This Web service is hosted by the ITL and by the Registry.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
This Web service calls the Preliminary_Checks function to validate the incoming message and add the message to the queue.
Outputs
CheckResponseObject
Call(s)
Preliminary_Checks

611

612 **Figure E35: ReceiveAuditTrail (Web service)**
613

Purpose
This is the HTTP SOAP request that registries use to send a transaction history for inconsistent blocks. If the SOAP request is not accepted, an HTTP SOAP fault is returned. Once the request has passed reconciliation message checks and been written to the queue, an HTTP SOAP response is returned. This Web service is hosted by the ITL.
Inputs
ReconciliationObject, TransactionObject
Table(s)
This function does not interact with the database.
Process
This function calls the Preliminary_Checks function to validate the incoming message and write it to the queue.
Outputs
CheckResponseObject
Call(s)
Preliminary_Checks

614

615 **Figure E36: ReceiveReconciliationResult (Web service)**
616

Purpose
This is the HTTP SOAP request that the STL uses to inform the ITL that it has finished processing a reconciliation action. If the SOAP request is not accepted, an HTTP SOAP fault is returned. Once the request has passed reconciliation message checks and been written to the queue, an HTTP SOAP response is returned. If necessary, the ITL relays the message to the appropriate registry. This Web service is hosted on the ITL and the Registry.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
This Web service calls the Preliminary_Checks function to validate the incoming message and add the message to the queue.
Outputs
CheckResponseObject
Call(s)
Preliminary_Checks

617

618 **Figure E37: ReceiveTotals (Web service)**
619

Purpose
This is the HTTP SOAP request that registries use to send the ITL the total number of units held by each account or account type at that registry. If the SOAP request is not accepted, an HTTP SOAP fault is returned. Once the request has passed reconciliation message checks and been written to the queue, an HTTP SOAP response is returned. This Web service is hosted on the ITL.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
This function calls the Preliminary_Checks function to validate the incoming message and write it to the queue.
Outputs
CheckResponseObject
Call(s)
Preliminary_Checks

620

621 **Figure E38: ReceiveUnitBlocks (Web service)**
622

Purpose
This is the HTTP SOAP request that registries use to send the total unit blocks held by each account or account type at that registry to the ITL. If the SOAP request is not accepted, an HTTP SOAP fault is returned. Once the request has passed reconciliation message checks and been written to the queue, an HTTP SOAP response is returned. This Web service is hosted on the ITL.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
This function calls the Preliminary_Checks function to validate the message and add it to the queue.
Outputs
CheckResponseObject
Call(s)
Preliminary_Checks

623

624 **Figure E39: Reconciliation_Data_Integrity_Check (Function)**
625

Purpose
This function will check the data in the ReconciliationObject and TransactionObject as appropriate to assure that it meets the minimum requirements to begin processing the incoming message. See Section 6.2.4 for a list of all data integrity checks performed by this function.
Inputs
ReconciliationObject, TransactionObject
Table(s)
Response_Catalog, Response_Log
Process
For each check identified in Section 6.2.4 evaluate the ReconciliationObject against the check. For each check, collect the appropriate response code returned and store in the CheckResponseObject.
Outputs
CheckResponseObject
Call(s)

626

627 **Figure E40: Reconciliation_Failure_Notification (Function)**
628

Purpose
The ITL calls this function to notify a registry or the STL that a reconciliation message sent by that party failed one or more message validation check.
Inputs
ReconciliationObject, CheckResponseObject
Table(s)
Process
Evaluate from where the message originated. If it was an STL, call Send_Reconciliation_Notification_To_STL. If it was a registry, call Send_Reconciliation_Notification_To_Registry.
Outputs
ResultIdentifier ReconciliationObject CheckResponseObject
Call(s)
Send_Reconciliation_Notification_To_STL Send_Reconciliation_Notification_To_Registry

629

630 **Figure E41: Reconciliation_Message_Sequence_Check (Function)**
631

Purpose
This function will check the data in the ReconciliationObject to assure that the reconciliation status identified is the expected sequence order. See Sections 6.2.5 and 6.2.6 for specifics on the sequence checks.
Inputs
ReconciliationObject
Table(s)
Response_Catalog, Response_Log
Process
For each check identified in Sections 6.2.5 and 6.2.6 evaluate the ReconciliationObject against the check. Modify the result indicator in the CheckResponseObject to indicate the success or failure of a check.
Outputs
CheckResponseObject
Call(s)

632

633 **Figure E42: Remove_From_Message_Queue (Function)**
634

Purpose
This function will remove a message from the processing queue when processing of the original message has completed without error.
Inputs
Table(s)
This function does not interact with the database.
Process
For the message being processed, remove message from the processing queue.
Outputs
Result_Identifier
Call(s)

635

Figure E43: Replace_Unit_Block (Function)

Purpose
This function will insert a new record in the Replacement_Unit_Block table in order to establish a relationship between the units being replaced and the replacing units.
Inputs
TransactionObject
Table(s)
Replacement_Unit_Block
Process
Retrieve the Replaced Block_ID for the block(s) to be replaced. Note that for a replacement transaction, it should not be necessary to split this block. Select Block_ID from the Unit_Block table where: • Holding_Registry_Code = replaced units registry code • Account_Type_Code = replaced units account type • Start_Block = replaced units start block • End_Block = replaced units end block Retrieve the Replacing Block_ID for the block(s) that replaces the old block. Select Block_ID from the Unit_Block table where: • Holding_Registry_Code = replacing units registry code • Account_Type_Code = replacing units account type • Start_Block = replacing units start block • End_Block = replacing units end block Append to the Replacement_Unit_Block table as follows: • Block_id = Replaced Block_ID • Replacement_Block_ID = Replacing Block_ID • Replacement_DateTime = current time
Outputs
Result_Identifier
Call(s)

638 **Figure E44: Request_Audit_Trail (Function)**
639

Purpose
This function will call the ProvideAuditTrail Web service to request a transaction history for specified blocks.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
Call Write_To_Message_Log Call ProvideAuditTrail Web service of the registry with parameters from the ReconciliationObject.
Outputs
ReconciliationObject
Call(s)
Write_To_Message_Log

640

641 **Figure E45: Request_Totals_From_Registry (Function)**
642

Purpose
This function will call the registry's Web service to request unit/type account type totals for reconciliation. If the "by Account" parameter is passed, then data are to be grouped by AccountIdentifier.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
Call Write_To_Message_Log Call ProvideTotals Web service of the registry with parameters set in the ReconcilationObject to request the unit holding count by account type and unit type.
Outputs
ReconciliationObject ResponseObject
Call(s)
Write_To_Message_Log

643

644 **Figure E46: Request_Unit_Blocks_From_Registry (Function)**
645

Purpose
This function will call the registry's Web service to request unit blocks for reconciliation for an account type and unit type.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
Call Write_To_Message_Log Call ProvideUnitBlocks Web service of the registry with parameters set in the ReconciliationObject.
Outputs
CheckResponseObject
Call(s)
Write_To_Message_Log

646

Figure E47: Retrieve_Message_From_Queue (Function)

Purpose
This function retrieves the first message in the queue to begin processing the next set of checks.
Inputs
Table(s)
Process
Retrieves the oldest time stamped item in the queue and stores it to a transaction object in preparation for further checks. The date and time that the message is placed in the message queue is the official date and time of record for the transaction or reconciliation message or status.
Outputs
TransactionObject ReconciliationObject
Call(s)
Write_To_Message_Log

650 **Figure E48: Reversal_of_Storage (Function)**
651

Purpose
Upon receiving instructions from the CDM Executive Board, this function will freeze all units associated with a project. It will then calculate the number of units each registry must retire due to a reversal of greenhouse gas storage for a project. This function will then inform each affected registry of the number of units that must be cancelled or replaced.
Inputs
Project ID, Percentage
Table(s)
Unit_block, Notification, Registry_Notification, Project
Process
Set freeze flag for all units associated with the project. Freeze all lCERs for that project Update Unit_Block Set Project_Feeze_Flag = 1 Where Project_ID = input project Id If any records found, insert record into the Notification table for "Reversal in Storage – Initial Notification" (type 5). Identify the registries holding affected units and calculate the number of units each registry must replace or cancel. Select Holding_Reg_Code, sum(Block_End - Block_Start + 1) as RegistrylCERTotal From Unit_Block Where Project_ID = input Project ID Group By Holding_Registry_Code For each record returned in the above resultset, calculate the number of lCERs to unfreeze at each registry. Number of lCERs to cancel or replace = input Percentage * RegistrylCERTotal If number of lCERs to unfreeze not an integer, round up to next integer. Insert record into Registry_Notification table with the number of lCERs to cancel or replace.

(cont.)

Figure E48: Reversal_of_Storage (Function) (cont.)

Process	
	Set Response Code to 6003 – lCERs may not be traded because the project associated with the units had a reversal in greenhouse gas storage. Each registry must cancel or replace a percentage of the units at that project before trading can resume. Compose warning message. The message begins with the following text: Notification ID #: The following number of lCERs must be cancelled or replaced because the project associated with the units had a reversal in greenhouse gas storage. Write number of lCERs to be canceled or replaced to the warning message. Call AcceptMessage with warning message on Holding Registry.
Outputs	
Result_Identifier	
Call(s)	

652

653 **Figure E49: Reversal_of_Storage_Replacement (Function)**
654

Purpose
This function will check for registry replacements due to the reversal in storage notification. If an adequate number of lCERs have been cancelled or replaced, the remaining lCERs held at that registry for the project will be unfrozen.
Inputs
Table(s)
Registry_Notification, Notification, Transaction_Log, Transaction_Block, Transaction_Log_History, Unit_Block
Process
Find open notifications to registries for reversal in storage Select Total_Units From Registry_Nofication join Notification on Notification_ID Where Notification_Type_Code = 5 And Registry_Notification_Date is null For each record found evaluate compliance by searching for transactions from the registry that reference this Notification ID. Select Notification_ID, sum(Block_End - Block_Start + 1) as RegistryUnits From Registry_Notification, join Transaction_Log, join Transaction_Block, Transaction_Log_History Where Notification_ID = input notification id And Transaction_Log_History.Transaction_Status_Code = “Completed” (4) And Transaction_Log_History.Transaction_Status_DateTime = most recent status for this transaction Group By Notification_ID

(cont.)

Figure E49: Reversal_of_Storage_Replacement (Function) (cont.)

Process
If RegistryUnits >= Total_Units, The registry has replaced all required units associated with this notification. Unfreeze remaining units at that project still held by this registry. Update Unit_Block Set Project_Freeze_Flag = 0 Where Project ID = input project ID And Holding_Registry_Code = input registry code Set the Registry_Notifcation date to mark the registry as having completed replacement. Update Registry_Notification Set Registry_Notification_Date = current date Where Notifcation ID = input notification ID And registry code = input registry code
Outputs
Result_Identifier
Call(s)

655

Figure E50: Send_Notification_To_Registry (Function)

Purpose
This function sends a HTTP SOAP request with the contents of the TransactionObject and the ResponseObject to a registry. This function is used to send notification to the Transferring Registry that either the proposal could not be processed due to discrepancies or no discrepancies were found. It is also used to notify the Acquiring Registry in an External transaction when a discrepancy is found in the Initiating Registry's proposal.
Inputs
TransactionObject, CheckResponseObject, RegistryCode
Table(s)
Process
Invoke the AcceptNotification Web service at the appropriate registry with the contents of the TransactionObject and the CheckResponseObject. This function waits for a HTTP SOAP response from the registry indicating the message was received successfully, at which time the request is logged to the Message_Log table.
Outputs
ResultIdentifier
Call(s)
Write_To_Message_Log

659 **Figure E51: Send_Proposal_To_Registry (Function)**
660

Purpose
This function will call the AcceptProposal Web service on the Acquiring Registry and add an entry in the message log.
Inputs
TransactionObject
Table(s)
Invoke the AcceptProposal Web service at the appropriate registry with the contents of the TransactionObject. This function waits for a HTTP SOAP response from the registry indicating the message was received successfully, at which time the request is logged to the Message_Log table.
Outputs
Result_Identifier
Call(s)
Write_To_Message_Log

661

662 **Figure E52: Send_Reconciliation_Notification_To_Registry (Function)**
663

Purpose
This function will call the registry's Web service to receive updates about an ongoing reconciliation action.
Inputs
ReconciliationObject CheckResponseObject
Table(s)
This function does not interact with the database.
Process
Call Write_To_Message_Log Call ReceiveReconciliationResult Web service on the registry.
Outputs
ReconciliationObject CheckResponseObject
Call(s)
Write_To_Message_Log

664

665 **Figure E53: Send_Reconciliation_Notification_To_STL (Function)**
666

Purpose
This function call the ReceiveReconciliationResult Web service on the STL.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
Call Write_To_Message_Log Call ReceiveReconciliationResult Web service on an STL.
Outputs
ReconciliationObject CheckResponseObject
Call(s)
Write_To_Message_Log

667

668 **Figure E54: Send_Reconciliation_Snapshot_DateTime (Function)**
669

Purpose
This function will call the InitiateReconciliation Web service on an STL in order to inform the STL of the snapshot DateTime of an upcoming reconciliation for a member registry.
Inputs
ReconciliationObject
Table(s)
This function does not interact with the database.
Process
Call Write_To_Message_Log Call InitiateReconciliation Web service method on destination STL.
Outputs
Result_Identifier
Call(s)
Write_To_Message_Log

670

671 **Figure E55: Send_To_STL (Function)**
672

Purpose
This function sends an HTTP SOAP request with the contents of the TransactionObject to an STL and logs the action to the Routing_Log.
Inputs
TransactionObject
Table(s)
Routing_Log
Process
This function will determine which Web service to invoke on the STL. If the transaction status = "Proposed" then the AcceptProposal will be called, otherwise the AcceptNotification will be called. This function waits for an HTTP SOAP response from an STL after which it writes a record to the Routing_Log indicating a transaction was forwarded to an STL for further processing.
Outputs
Result_Identifier
Call(s)
Write_To_Routing_Log

673

674 **Figure E56: Snapshot_Registry_Data (Function)**
675

Purpose
This function will take a "snapshot" of the holdings in the unit_block table for a specified registry. This function will be called precisely at the snapshot time and will retrieve all the unit blocks held at a specified registry. In addition to the identifying information for the unit block, the account type and unit type are also retreived. The ITL will perform reconciliation validation operations on the resultant dataset.
Inputs
ReconciliationObject
Table(s)
Unit_Block
Process
Select Holding_Registry_Code, Account_Type_Code, Unit_Type_Code, Originating_Registry, Start_Block, End_Block From Unit_Block Where Holding_Registry_Code = code for registry subject to reconciliation The returned dataset will be inserted into a snapshot unit table for reconcilation validation processes.
Outputs
Result_Identifier
Call(s)

676

677 **Figure E57: Split_Block (Function)**
678

Purpose
This function will search the the Unit_Block table for a block involved in a transfer, and split the existing block if necessary so that only the units involved in a transaction will be affected. The new blocks created will have the same characteristics (account type, block type, etc.) as the parent block. There are four scenarios to account for: • Block to be transferred matches serial numbers exactly • Begin numbers match • End numbers match • Neither numbers match, but new block needs to be created.
Inputs
RegistryUnitBlockObject
Table(s)
Unit_Block
Process
Step 1: Search the Unit_Block table for records that exactly match the block to be transferred. Search for: • Originating_Registry_Code = input originating registry code • Start_Block = input start block • End_Block = input end block If found, there is no need to split the block. Store Block_ID to array of Block ID. Function ends. If not found, proceed to Step 2 Step 2: Search the Unit_Block table for a single record where the starting serial numbers match, and the end number in the Unit_Block table is greater than the end number of the block to be transferred. Search for: • Originating_Registry_Code = input originating registry code • Start_Block = input start block • End_Block > input end block

(cont.)

Figure E57: Split_Block (Function) (cont.)

Process
If found, split the block into two blocks by modifying the beginning number of existing block in the Unit_Block table (Set Start_Block = input end block + 1) insert record in Unit_Block table for the block to be transferred (Call Split_Related_Blocks) Store Block IDs to array of Block ID If not found, proceed to Step 3 Step 3: Search the Unit_Block table for a single row where the end serial number matches the end number of the block to be traded, and the Unit_Block start number is less than the start number of the block to be traded. Search for: • Originating_Registry_Code = input registry code • Start_Block < input start block • End_Block = input end block If found, split the block into two blocks by modifying the beginning number of existing block in the Unit_Block table (Set End_Block = input start block– 1) insert record in Unit_Block table for the new block (Call Split_Related_Blocks) Store Block_IDs to array of Block ID If not found, proceed to Step 4 Step 4: Search Unit_Block table for a single row where the block to be transferred is completely contained by an existing block. Search for: • Originating_Registry_Code = input originating registry code • Start_Block < input start block • End_Block > input end block

(cont.)

Figure E57: Split_Block (Function) (cont.)

Process
If found, split the block into three blocks by modifying the ending number of existing block in the Unit_Block table (Set End_Block = input Start Block - 1) Insert record in Unit_Block table for the new block to be transferred Call Insert_Unit_Block with input Start_Block and input End_Block. Insert record in Unit_Block table where Start_Block = input End_Block + 1 and End_Block = original End_Block Call Split_Related_Blocks Store Block_IDs to array of Block IDs If not found, continue to Step 5 Step 5: Search the Unit Block table for the block that contains the following: Create record set of Unit_Block where • Originating_Registry_Code = input originating registry code • Maximum (Start_Block) ≤ input start block This is Block A. Split Block A into two blocks by: Modifying the ending number of existing block in the Unit_Block table (set End_Block = input start block - 1) Insert record into Unit_Block table for the part of the block to be transferred Call Insert Unit Block with input start block and Block A's original end number Call Split_Related_Blocks Add new block to Block A record set and remove original Block_ID Create record set of Unit_Block where • Originating_Registry_Code = input originating registry code • Minimum (End_Block) ≥ input end block This is Block B. Split the block into two blocks by: Modifying the starting number of existing block in the Unit_Block table (set Start_Block = input end block + 1) Insert record into Unit_Block table for the part of the block to be transferred Call Split_Related_Block Add new block to Block B record set and remove original Block_ID Create record set Block C where Block_IDs are between record set A and B. Create instance of ITLUnitBlockObject for every unique Block_ID in Block A + B + C record sets

(cont.)

Figure E57: Split_Block (Function) (cont.)

Outputs
Result Identifier, ITLUnitBlockObject
Call(s)
Write_To_Inconsistent_Block

679

680 **Figure E58: Split_Related_Blocks (Function)**
681

Purpose
This function splits related blocks in the Replacement_Unit_Block table and the Inconsistent_Block table when a split has been made in the Unit_Block table.
Inputs
Original_Block_ID, New_Block_ID
Table(s)
Inconsistent_Block Replacement_Unit_Block
Process
Select from Inconsistent_Block table where Block_ID = Original_Block_ID. If found, insert new record where • Recon_Log_ID = Recon_Log_ID • Initiating_Registry_Code = Initiating_Registry Code • Block_ID = New_Block_ID Select from Replacement_Unit_Block table where Replacement_Block_ID = Original_Block_ID If found, insert new record where • Replacement_Block_ID = New_Block_ID • Block ID = Block_ID • Replacement_DateTime = Replacement_DateTime Note that the Block_ID does not split as the Replacement_Block_ID has. The total units of all the Replacement_Block_IDs associated with the one Block_ID should be equal.

682

683 **Figure E59: Start_Reconciliation (Function)**
684

Purpose
This function opens a reconciliation action and requests the registry subject to reconciliation to send data to the ITL. The data requested will depend on the phase in which reconciliation was requested to start. Prior to opening a new reconciliation, this function will check if there is an ongoing reconciliation action for the specified registry. If there is, the reconciliation initiation will be denied and the STL or ITL manager will be notified if it requested the reconciliation.
Inputs
ReconciliationObject Phase
Table(s)
Reconciliation_Log
Process
Generate a new reconciliation ID and Call Write_To_Reconciliation_Log to record the new reconciliation action. Verify that another reconciliation action for this registry is not ongoing. Query the Reconciliation_Log table to identify any reconciliation action without an end date. Call Write_To_Reconciliation_Status to update the status to 9 ("Reconciliation Start Denied"). Call Close_Reconciliation_Action to update the end date in the Reconciliation log table. The Response Code is 4001. If the source of the reconciliation request was an STL, call Send_Reconciliation_Notification_To_STL with the response code. Call Trade_Scheme_Member to determine if the registry is part of an STL. If the registry is part of an STL, call Send_Reconciliation_Snapshot_DateTime. If no ongoing reconciliation is found, call Write_To_Reconciliation_Status with 1 ("Initiated"). If requested phase = 1, call Request_Totals_From_Registry. If requested phase = 2, call Request_Unit_Blocks_From_Registry. If requested phase = 3, call Request_Audit_Trail_From_Registry.

(cont.)

Figure E59: Start_Reconciliation (Function) (cont.)

Outputs
CheckResponseObject
Call(s)
Write_To_Reconciliation_Log Write_To_Reconciliation_Status Request_Totals_From_Registry Request_Unit_Blocks_From_Registry Request_Audit_Trail_From_Registry Send_Reconciliation_Notification_To_STL Close_Reconciliation_Action

685

686 **Figure E60: Time_Sync (Function)**
687

Purpose
This function will run on a period basis to check the system time of the registries that are a part of the trading network.
Inputs
RegistryCode
Table(s)
System_Log
Process
Call ProvideTime web service method on the input registry. Insert new record in the System_Log table with • Registry Code • Registry Date and Time • ITL Date and Time
Outputs
Result_Identifier
Call(s)

688

689 **Figure E61: Trade_Scheme_Member (Function)**
690

Purpose
This function identifies whether the registry is a member of a specified trading scheme.
Inputs
Registry_Code, Trade_Scheme
Table(s)
Registry_Trading_Scheme
Process
Query the Registry_Trading_Scheme table to see if the Registry_Code and Trade_Scheme exist. If so, return true (1).
Outputs
Result_Identifier
Call(s)

691

692 **Figure E62: Transaction_Cleanup (Function)**
693

Purpose
This function runs every hour and checks for transactions that have not completed and for which the last message was initiated more than 24 hours ago. If found, the transaction is Cancelled.
Inputs
Table(s)
Transaction_Log, Transaction_Log_History
Process
In progress transactions will have a status of: • 1 - Proposed • 2 - Checked (No Discrepancy) • 3 - Checked (Discrepancy) • 6 - Rejected • 8 - Accepted • 9 - STL Checked (No Discrepancy) • 10 - STL Checked (Discrepancy) Every Hour: Select Transaction_ID, Transaction_Status_ID From Transaction_Log, Transaction_Log_History Where Transaction_Status_ID = (1,2,3,6,8,9,10) Transaction_Status_DateTime = most recent status for that Transaction_ID and Current Time – Transaction_Status_DateTime > 24 hours For each record found, Set the Response Code to 6000 – A transaction must be completed within 24 hours of initiation. This transaction has exceeded that limit and has been cancelled. Call Write_Transaction_Status with Transaction_ID and “Cancelled” (7) For each Unit Block involved in the transaction, call Delineate_Units with 0 so the units may be involved in another transaction.

(cont.)

Figure E62: Transaction_Cleanup (Function) (cont.)

Process
Call Send_Notification_To Registry for the Initiating Registry. If Transaction Type is External (type 3), Call Send_Notification_To_Registry for the Acquiring Registry. Call Trade_Scheme_Member for the Initiating Registry and the Acquiring Registry (if applicable). If either registry is part of an STL, call Send_To_STL for the identified STL.
Outputs
Result_Identifier
Call(s)
Write_Transaction_Status Delineate_Units Send_Notification_To_Registry Send_To_STL

694

695 **Figure E63: Update_Block_Ownership_Or_Account_Type (Function)**
696

Purpose
This function will update the Holding_Registry_Code and Account_Type attributes of the Unit_Block table in order to complete a unit block transfer.
Inputs
TransactionObject, ITLUnitBlockObject
Table(s)
Unit_Block
Process
For each Block ID in the ITLUnitBlockObject, update the following: • Holding_Registry_Code = acquiring registry code • Account_Type_Code = acquiring account type
Outputs
Result_Identifier
Call(s)

697
698

Figure E64: Update_Unit_Block (Function)

Purpose
This function will update a record in the Unit_Block table. The only fields that this function updates are the Holding_Registry_Code, Account_Type_Code, Unit_Type_Code, the Applicable_Commitment_Period, and Supplementary_Unit_Type.
Inputs
ITLUnitBlockObject, RegistryUnitBlockObject
Table(s)
Unit_Block
Process
If the transaction type is External (3), update the Holding_Registry_Code and Account_Type_Code to the registry and account type of the acquiring party. If transaction type is Carry-over (4), update the Applicable_Commitment_Period to the current commitment period. If the transaction type is Conversion (2), and the unit was not previously (2) RMU, update the Unit_Type to (3) for ERU. Update the Project Identifier. If the transaction type is Conversion (2), and the previous unit type was (2) RMU, update the Unit_Type to (4) for ERU converted from RMU. Update the Project Identifier. If the transaction type is Cancellation (4) or Retirement (5), update the Account_Type to the account type of the acquiring party. If the transaction type is Expiration Date Change (8), update the Expiration Date. If the transaction type is Internal/Supplementary (10) and the Supplementary Transaction Code=52 then the Supplementary_Unit_Type is updated to the appropriate code for the new Supplementary Transaction Type Code.
Outputs
Result_Identifier
Call(s)

Figure E65: Validate_Proposal (Function)

Purpose
This function will enter a transaction in the ITL, evaluate the transaction based on its transaction type, accumulate all the appropriate responses, update the transaction status, and freeze the units involved in the transaction.
Inputs
TransactionObject
Table(s)
This function does not interact with the database.
Process
Set Select for update on database transaction. Call Delineate_Units with 1 to indicate that the unit blocks are involved in a transaction. Commit database transaction. Set Select for Update on database transaction. Record the transaction by calling the Write_Transaction procedure followed by Write_Transaction_Block for each unit block involved in the transaction. Set the transaction status to "Proposed" (1) by calling Write_Transaction_Status. Once recorded, validate the transaction against the business rules. Call Get_Checks, followed by Execute_Checks to retrieve and execute the business checks for this transaction. Get_Checks is first called to return checks that are general and apply to all transactions, and then again to return checks that are specific to the current transaction type. As each unit block is evaluated, call Write_Block_History to record the problem if a check is failed. If any of the checks fail, call Write_Transaction_Status with 3 for "Checked (Discrepancy)." If no checks fail, call that same function with 2 for "Checked (No Discrepancy)." Finally, no matter what the Transaction Status, call Delineate_Units with 1, to set the Unit_Status in the Transaction_Block table and prevent the units from being involved in another transaction until this transaction is complete or the discrepancy is resolved. Commit database transaction.

(cont.)

704 **Figure E65: Validate_Proposal (Function) (cont.)**
705

Outputs
Result_Identifier, CheckResponseObject
Call(s)
Write_Transaction Write_Transaction_Block Write_Transaction_Status Get_Checks Execute_Checks Write_Block_History Delineate_Units

706

707 **Figure E66: Validate_Totals (Function)**
708

Purpose
This function will compare each unit block sent by the registry against the Snapshot of the ITL records previously taken. If blocks do not match, they will be marked as inconsistent.
Inputs
Table(s)
Process
For each item in the TotalsObject array, calculate the number of units held by that account type and unit_type. Select Sum(end_block - start_block + 1) as ITL Totals From Reconcilation Snapshot Table Where Reconciliation_ID = current reconciliation action and Account_Type = TotalsObject Account Type and Unit_Type = TotalsObject Unit_Type If TotalsObject Totals <> ITL Totals, call Write_To_Reconciliation_Status with 3 ("Totals Inconsistent"). Loop to next item in TotalsOjbect array If any of the totals did not match, Continue to next step of reconciliation set response code = 6010 call Request_Unit_Blocks_From_Registry If all totals match, call Write_To_Reconciliation_Status with 2 ("ITL Validated"). call Trade_Scheme_Member to determine if registry is part of supplementary program. If part of supplementary program, call Request_Totals_By_Account

(cont.)

Figure E66: Validate Totals (Function) (cont.)

Process
else Notify Registry of successful reconciliation call Send_Reconciliation_Notification_To_Registry Clean up old Inconsistent Blocks, if any, now that a good reconciliation has been performed. Select Inconsistent_Block_ID From Inconsistent_Block Where Originating_Registry_Code = code for registry subject to this reconciliation action For each item found, Call Delete_Inconsistent_Block
Outputs
Call(s)
Write_To_Reconciliation_Status Request_Unit_Blocks_From_Registry Trade_Scheme_Member Request_Totals_By_Account Delete_Inconsistent_Block Send_Reconciliation_Notification_To_Registry

709
710

711 **Figure E67: Validate_Unit_Blocks (Function)**
712

Purpose
This function will compare each unit block sent by the registry against the ITL records. If blocks do not match, they will be marked as inconsistent.
Inputs
ReconciliationObject
Table(s)
Unit_Block
Process
To simplify the comparison, this function will first analyze the UnitBlockObject array within the ReconciliationObject and recombine all unit blocks with adjacent serial numbers. The reduced unit blocks will be stored in a new Combined UnitBlockObject array. The registry, account type, and unit_type is listed for each unit block. Likewise, this function will analyze the Unit_Block table and combine unit blocks with adjacent serial numbers and held in the same account type at the registry in question. These unit blocks will be stored in the Master UnitBlockObject array. The Master UnitBlockObject will maintain a record of the original block id of the unit. If multiple blocks were combined to create the record, the id of each block combined will be stored. Create a new field to serve as a key field. This field should be the account type combined with the block start number. Order each array by the key field. Loop through the arrays, keeping them in sync, and compare the corresponding records of each file until the end of both arrays is reached. In normal circumstances, the two reduced unit block arrays will be identical and no inconsistent blocks will be identified. Loop until the end of both the Combined and Master array is reached. Compare Records If Combined Key field = Master Key Field, There is no problem. Move to next record in Combined Array. Move to next record in Master Array. If Combined Key Field < Master Key Field OR End of Master Array, The Combined array has a key that is not in the Master Array. Call Get_Block_ID to return any overlapping blocks in the Unit_Block table

(cont.)

713 **Figure E67: Validate_Unit_Blocks (Function) (cont.)**
714

Process
For each Block_ID returned, Call Insert_Inconsistent_Block Move to next record in Combined Array. Do not move to next record in Master array. If Combined Key Field > Master Key Field or End of Combined Array, The Master Array has a key that is not in the Combined Array. Call Insert_Inconsistent_Block for each block id that made up the record in the Master Array unit block. Move to next record in the Master Array. Do not move to next record in Combined Array. If any inconsistent blocks are found, call Write_To_Reconciliation_Status to update the status to 4 ("Unit Blocks Inconsistent"). Call Request_Audit_Trail to have the registry send a transaction history since the last successful reconciliation for each inconsistent block. If no inconsistent blocks are found, the reconciliation enters the manual intervention phase in order to explain why the unit total counts did not match in the first step of reconciliation. Move to next record in Combined Array. Do not move to next record in Master array. If Combined Key Field > Master Key Field or End of Combined Array, The Master Array has a key that is not in the Combined Array. Call Insert_Inconsistent_Block with the Master Array unit block. Move to next record in the Master Array. Do not move to next record in Combined Array. If any inconsistent blocks are found, call Write_To_Reconciliation_Status to update the status to 4 ("Unit Blocks Inconsistent"). Call Request_Audit_Trail to have the registry send a transaction history since the last successful reconciliation for each inconsistent block. If no inconsistent blocks are found, the reconciliation enters the manual intervention phase in order to explain why the unit total counts did not match in the first step of reconciliation.
Outputs
Result_Identifier ResponseObject
Call(s)
Insert_Inconsistent_Block Request_Audit_Trail Write_To_Reconciliation_Status Get_Block_ID

715

716 **Figure E68: Write_Audit_Trail_To_File (Function)**
717

Purpose
This function will create a new text file and transfer the transaction history presented in the incoming message to the newly created file. The transaction history will be stored in XML format.
Inputs
TransactionObject
Table(s)
Process
Create and open new text file. For each transaction in the TransactionObject array, write contents to text file.
Outputs
Result_Identifier File Name and Path
Call(s)

718

719 **Figure E69: Write_Block_History (Function)**
720

Purpose
This function will insert a record in the Transaction_Block_History table.
Inputs
Transaction_Block_ID, Response_Code
Table(s)
Transaction_Block_History
Process
Append to the Transaction_Block_History table the DateTime and response code passed for a Transaction_Block_ID.
Outputs
Result_Identifier
Call(s)

721

722 **Figure E70: Write_To_File (Function)**
723

Purpose
This function will create a text file, write the contents of the HTTP SOAP Request to the file, then add the file to a master Zip file.
Inputs
Web_Service_URL, Transaction_Type, XML Message Content, File_name
Table(s)
This function does not interact with the database.
Process
Retrieve the to and from elements in the contents of the SOAP request. Retrieve the Registry Code and Transaction Identifier values. Generate the file name by concatenating these two values along with a random number. Write contents of XML body to text file and store in the Zip file. If the file fails to write to file, return HTTP SOAP response error. Record file name and the Zip file name in which the file is compressed to message queue to be recorded in Message_Log.
Outputs
Result_Identifier
Call(s)

724

Figure E71: Insert_Unit_Block (Function)

Purpose
This function will insert a row into the Unit_Block table when a new unit block is created.
Inputs
TransactionObject, ITLUnitBlockObject
Table(s)
Unit_Block
Process
Append a record into the Unit_Block table for the unit block in the ITLUnitBlockObject associated with the TransactionObject.
Outputs
Result_Identifier
Call(s)

728 **Figure E72: Write_To_Message_Log (Function)**
729

Purpose
This function will append a record to the Message_Log. The Message_Log table records the history of all ITL message exchange. The contents of an HTTP SOAP Request received from either a registry, CDM or the STL are parsed and constructed as text files that are then stored in a larger Zip file. The Message_Log tracks the time when the file was created, the name of the file and the name of the Zip file in which it has been compressed.
Inputs
Filename, Master_File_Name, Registry_Code, Reconciliation ID, Transaction ID, Web service
Table(s)
Message_Log
Process
Append to Message_Log, Registry_Code, File_Name, Master_File_Name, SystemTime( ) Reconciliation ID or Transaction ID, Web service to Registry_Code, File_Name, File_Path, Submission_Date, Recon_ID, Transaction_ID, Web_Service.
Outputs
Result_Identifier
Call(s)

730

731 **Figure E73: Write_To_Message_Queue (Function)**
732

Purpose
This function adds the HTTP SOAP request information to a message queue.
Inputs
Include all incoming HTTP SOAP request information.
Table(s)
This function does not interact with the database.
Process
This function adds the contents of all incoming requests to the appropriate message queue. In addition, the URL of the Initiating Registry and the date and time of the request is added to the queue. Note that the date and time a message is added to the queue becomes the official date and time of record for the transaction.
Outputs
Result_Identifier
Call(s)

733

734 **Figure E74: Write_To_Reconciliation_Log (Function)**
735

Purpose
This function inserts a new record into the Reconciliation_Log table.
Inputs
ReconciliationObject
Table(s)
Reconciliation_Log
Process
Append to Reconciliation_Log table as follows: • Recon_ID = input reconciliation ID • Recon_Action BeginDateTime = input reconciliation begin DateTime • Registry_Code = input registry • Recon_Log_Comment = null • Recon_Action EndDateTime = input reconciliation end DateTime • Recon_Snapshot_DateTime = input reconciliation snapshot DateTime • Recon_Phase_Code = input reconciliation phase
Outputs
ResponseObject
Call(s)

736

737 **Figure E75: Write_To_Reconciliation_Status (Function)**
738

Purpose
This function inserts a new record into the Reconciliation_Status_History table.
Inputs
ReconciliationObject
Table(s)
Reconciliation_Status_History
Process
Append to Reconciliation_Status_History table as follows: • Recon_ID = input reconciliation ID • Recon_Status_Code = input reconciliation status code • Recon_Log_DateTime = system time • Recon_Comment = null
Outputs
ResponseObject ReconciliationObject
Call(s)

739

740 **Figure E76: Write_To_Routing_Log (Function)**
741

Purpose
This function will add a record to the Routing Log in order to track interactions with an STL.
Inputs
Route_Status_Code, Registry_Code, Transaction_ID
Table(s)
Routing_Log
Process
Add record to Routing Log indicating the TransactionObject (or any other object) has been sent to an STL or has been received from an STL.
Outputs
Call(s)

742

743 **Figure E77: Write_Transaction (Function)**
744

Purpose
This function will insert a record into the Transaction_Log table.
Inputs
TransactionObject
Table(s)
Transaction_Log
Process
Append to Transaction_Log the elements in the TransactionObject.
Outputs
Result_Identifier
Call(s)
Write_Transaction_Block

745

746 **Figure E78: Write_Transaction_Block (Function)**
747

Purpose
This function will insert a record into the Transaction_Block table.
Inputs
TransactionObject, RegistryUnitBlockObject
Table(s)
Transaction_Block
Process
For each instance of the RegistryUnitBlock associated with the TransactionObject, insert a record into the Transaction_Block table.
Outputs
Result_Identifier
Call(s)
Write_Transaction_Status

748

749 **Figure E79: Write_Transaction_Status (Function)**
750

Purpose
This function will insert a record into the Transaction_Log_History.
Inputs
TransactionObject
Table(s)
Transaction_Log_History
Process
Append a record to the Transaction_Log_History table with Transaction_ID, system time stamp and current transaction status code from TransactionObject.
Outputs
Transaction_Log_History_ID, unique identifier for new Transaction_Log_History record, Result_Identifier
Call(s)

751

Annex F
List of Transaction Checks

1. Version and Authentication Checks

Check Name	Certificate
Check Description	If certificate is not recognized, message is rejected.
Process	Certificate is checked by IPSec VPN against a Certificate Authority.
Response Code	None
Response Description	SOAP error.

Check Name	SOAP Identifier
Check Description	Initiating Registry must be consistent with sender of SOAP message.
Process	
Response Code	None
Response Description	SOAP error.

Check Name	WSDL Check
Check Description	Message must conform to WSDL.
Process	
Response Code	None
Response Description	SOAP error.

Check Name	Major Version
Check Description	If Major Version number in transaction message does not match Major Version number for DES.
Process	If input major version <> major version of DES, return response code 1031.
Response Code	1031
Response Description	Registry version is not current. Please see www.TransactionLogUpdates.Updates.htm for recommended upgrade elements.

Check Name	Minor Version
Check Description	If Minor Version number in transaction message does not match Minor Version number for DES.
Process	If input minor version <> minor version of DES, return response code 1032.
Response Code	1032
Response Description	Registry version is not current and is no longer compatible with ITL. Please see www.TransactionLogUpdates.Updates.htm for required upgrade elements.

2. Message Validity Checks

Check Name	Message Age
Check Description	Message must be processed within 24 hours of submission.
Process	If transaction DateTime - current DateTime > 24, return response code 1301.
Response Code	1301
Response Description	The message was held in the message queue for more than 24 hours. The message cannot be processed.

3. Registry Checks

Check Name	Identify Registry
Check Description	Must be contained in Registry table.
Process	Select Registry_Code From Registry Where Registry_Code = input registry code. If not found, return response code 1501.
Response Code	1501
Response Description	Registry code invalid or unknown. This message cannot be processed.

Check Name	Initiating Registry Available for Transactions
Check Description	Initiating Registry status code must be equal to zero.
Process	Select Registry_Status_Code from Registry_Status_History Where Registry_Status_Date = most recent registry status and Registry.Registry Code = Initiating Registry. If not found or Registry_Status_Code <> 0, return response code 1503.
Response Code	1503
Response Description	Initiating Registry is not currently authorised to propose transactions for validation.

779
780

Check Name	Acquiring Registry Available for Transactions
Check Description	Acquiring Registry status code must be equal to zero.
Process	Select Registry_Status_Code from Registry_Status_History Where Registry_Status_Date = most recent registry status and Registry.Registry Code = Acquiring Registry. If not found or Registry_Status_Code <> 0, return response code 1504.
Response Code	1504
Response Description	Acquiring Registry is not currently authorised to receive transfers. This message cannot be processed.

781
782

Check Name	Registry Available for Reconciliation Action
Check Description	Registry status code must be zero or 1.
Process	Select Registry_Status_Code from Registry_Status_History Where Registry_Status_Date = most recent registry status and Registry.Registry Code = Initiating Registry code. If not found or Registry_Status_Code <> 0 or 1, return response code 1510.
Response Code	1510
Response Description	Initiating Registry is not currently authorised to submit reconciliation data.

783
784

4. Data Integrity Checks

Check Name	Transaction Mask
Check Description	Transaction ID must be comprised of a registry code followed by numeric values
Process	Determine if first 2 or 3 characters are alpha. Determine if remaining characters are numeric. If not, return response code 2001.
Response Code	2001
Response Description	Transaction ID has invalid format. This message cannot be processed.

Check Name	Transaction Status Code
Check Description	Transaction status code must be valid.
Process	If input transaction status code < 1 or > 10, return response code 2003.
Response Code	2003
Response Description	Transaction status code invalid. This message cannot be processed.

Check Name	Transaction Type Code
Check Description	Transaction type must be (1) Issuance, (2) Conversion, (3) External, (4) Cancellation, (5) Retirement, (6) Replacement, (7) Carry-over, (8) Expiry Date Change or (10) Internal/Supplementary Program.
Process	Verify that the input transaction type is 1, 2, 3, 4, 5, 6, 7, 8 or 10. If not, return response code 2004.
Response Code	2004
Response Description	Transaction type code invalid. This message cannot be processed.

Check Name	ERU, CER, tCER, lCER Check
Check Description	If the unit is a CER, tCER, lCER, ERU, or an ERU converted from an RMU, a Project identifier must be present.
Process	If input unit type = (3, 4, 5, 6 or 7), If Input Project ID is blank, return response code 2005.
Response Code	2005
Response Description	Unit blocks of type ERU, CER, tCER or lCER must have a Project ID. This message cannot be processed.

794
795

Check Name	ERU Track
Check Description	If a unit is an ERU, a valid track must be present.
Process	Verify that track is equal to 1 or 2.
Response Code	2006
Response Description	Unit blocks of type ERU must have a track number. This message cannot be processed.

796
797

Check Name	Supplementary Transaction Type Code
Check Description	The Supplementary Transaction Type Code must be blank or valid.
Process	Verify that if a value exists for supplementary transaction type, that it is in the Supplementary Transaction Code table.
Response Code	2007
Response Description	Supplementary Transaction Type Code invalid. This message cannot be processed.

798
799

Check Name	Initiating Account Identifier
Check Description	Initiating Account ID must be greater than zero.
Process	If input account identifier is not ≥ zero, return response code 2080.
Response Code	2080
Response Description	Account identifier must be > zero. This message cannot be processed.

800
801

Check Name	Acquiring Account Identifier
Check Description	Acquiring account identifier must be greater than zero.
Process	If input account identifier for acquiring account is not ≥ 0, return response code 2082.
Response Code	2082
Response Description	Acquiring account identifier is equal to zero. This message cannot be processed.

802
803

Check Name	Account Type Code
Check Description	Account Type Code must be valid.
Process	If input account type not (100, 110, 120, 121, 210, 220, 230, 240, 300, 411, 421, 422, 423), return response code 2083.
Response Code	2083
Response Description	Account type invalid. This message cannot be processed.

804
805

Check Name	Unit Serial Range
Check Description	Unit Serial end block serial number must be greater than or equal to the Unit Serial begin block serial number.
Process	If input end block < input start block, return response code 2084.
Response Code	2084
Response Description	Serial Block invalid. This message cannot be processed.

806
807

Check Name	Unit Type Code
Check Description	Unit type must be valid.
Process	If input unit type < zero or > 7, return response code 2085.
Response Code	2085
Response Description	Unit type code invalid. This message cannot be processed.

808
809

Check Name	Unit Serial
Check Description	Unit Serial start block and Unit Serial end block must have a value greater than zero.
Process	If input block start ≤ zero or Input block end ≤ zero, return response code 2086.
Response Code	2086
Response Description	Serial number must be greater than zero. This message cannot be processed.

Check Name	Supplementary Unit Type Code
Check Description	Supplementary Unit Type must be valid.
Process	Verify that if a value exists for supplementary unit types, that it is in the Supplementary Unit Type Code table.
Response Code	2087
Response Description	Supplementary Unit Type Code invalid. This message cannot be processed.

Check Name	Notification ID/Registry
Check Description	The Notification ID must be one that was sent to the registry.
Process	If Notification ID and registry code do not exist in Registry Notification table, return response code 2089.
Response Code	2089
Response Description	A notification with this ID was not sent to the registry. This message cannot be processed.

Check Name	LULUCF Code
Check Description	LULUCF activity code must be valid.
Process	If LULUCF code <> (1, 2, 3 or 4), return response code 2090.
Response Code	2090
Response Description	LULUCF activity code invalid. This message cannot be processed.

Check Name	Commitment Period
Check Description	Original and Applicable Commitment Period must be less than 10.
Process	If input original Commitment Period is > 10 or If input applicable Commitment Period is > 10, return response code 2091.
Response Code	2091
Response Description	Commitment Period invalid. This message cannot be processed.

Check Name	Track Code
Check Description	Track must be blank, 1 or 2.
Process	If input track <> (1 or 2), return response code 2096.
Response Code	2096
Response Description	Track code is invalid. This message cannot be processed.

Check Name	Transaction Status DateTime
Check Description	Transaction Status DateTime must be between 1/1/05 and current date.
Process	If the input transaction status DateTime is not between 1/1/05 and the system date, return response code 2098.
Response Code	2098
Response Description	Transaction status date invalid. This message cannot be processed.

Check Name	RMU
Check Description	If a unit is an RMU, a valid LULUCF code must be present for the unit block.
Process	If input unit type = 2, If Input LULUCF Code is not (1, 2 or 3), return response code 2100.
Response Code	2100
Response Description	Units of type RMU must have a LULUCF code. This message cannot be processed.

826
827
828 ## 5. Message Sequence Checks for Transactions from Registries
829
830

Check Name	Transaction Number Does Not Exist
Check Description	Transaction ID does not exist and transaction status = "Terminated", "Completed", or "Accepted".
Process	If input transaction status code is (4, 5 or 8), Select Transaction_ID From Transaction_Log Where Transaction_ID = input Transaction_ID If not found, return response code 3001.
Response Code	3001
Response Description	The transaction number identified in the XML message does not exist in the ITL database. This message cannot be processed.

831
832

Check Name	Transaction Status Out of Sequence for Prior Completed Status
Check Description	Transaction status = "Completed" and prior transaction status = "Completed".
Process	If input transaction status code = 4, Select Transaction_Status_Code From Transaction_Log_History Where Transaction_ID = input transaction ID and Transaction_Status_DateTime = most recent status for that transaction and Transaction_Status_Code = 4 If found, return response code 3002.
Response Code	3002
Response Description	Transaction has already been completed. This message cannot be processed.

833
834

Check Name	Non-external Accepted Status
Check Description	Transaction status = "Accepted" for non-external transaction.
Process	If input transaction type <> 3 and If input transaction status = 8, return response code 3003.
Response Code	3003
Response Description	Transaction status of "Accepted" not valid for non-external transaction. This message cannot be processed.

835
836

Check Name	Transaction Status Not Valid
Check Description	Transaction status = "Checked (No Discrepancy)", "Checked (Discrepancy)", "STL Checked (No Discrepancy)", or "STL Checked (Discrepancy)".
Process	If input transaction status code = 2, 3, 9 or 10, return response code 3004.
Response Code	3004
Response Description	Transaction status not valid for registry message. This message cannot be processed.

837
838

Check Name	Transaction Status Out of Sequence for Prior STL Discrepancy Status
Check Description	Transaction status = "Completed" and prior transaction status = "STL Checked (Discrepancy)".
Process	If input transaction status code = 4; Select Transaction_Status_Code From Transaction_Log_History Where Transaction_ID = input transaction ID and Transaction_Status_Datetime = most recent status for that transaction and Transaction_Status_Code = 10 If found, return response code 3005.
Response Code	3005
Response Description	The transaction identified has already been recorded as an STL Discrepancy. This message cannot be processed.

839
840

Check Name	Transaction Status Out of Sequence for Prior Cancelled Status
Check Description	Transaction status = "Completed" and prior transaction status = "Cancelled".
Process	If input transaction status code = 4, Select Transaction_Status_Code From Transaction_Log_History Where Transaction_ID = input transaction id and Transaction_Status_DateTime = most recent status for that transaction and Transaction_Status_Code = 7 If found, return response code 3006.
Response Code	3006
Response Description	Transaction has already been cancelled. This message cannot be processed.

841
842

Check Name	Transaction Status Out of Sequence for Prior Terminated Status
Check Description	Transaction status = "Completed" and prior transaction status = "Terminated".
Process	If input transaction status code = 4, Select Transaction_Status_Code From Transaction_Log_History Where Transaction_ID = input transaction ID and Transaction_Status_DateTime = most recent status for that transaction and Transaction_Status_Code = 5 If found, return response code 3007.
Response Code	3007
Response Description	Transaction has already been terminated. This message cannot be processed.

843
844

Check Name	Transaction Status Out of Sequence for Prior Checked Discrepancy Status
Check Description	Transaction status = "Completed" and prior transaction status = "Checked (Discrepancy)".
Process	If input transaction status code = 4 and input transaction type <> 10, Select Transaction_Status_Code From Transaction_Log_History Where Transaction_ID = input transaction ID and Transaction_Status_DateTime = most recent status for that transaction and Transaction_Status_Code = 3 If found, return response code 3008.
Response Code	3008
Response Description	The transaction identified has already been recorded as a Discrepancy. This message cannot be processed.

845
846

Check Name	Transaction ID Not Unique
Check Description	Transaction status = "Proposed" and Transaction ID already exists.
Process	If input transaction status code = 1, Select Transaction_ID From Transaction_Log Where Transaction_ID = input transaction ID If found, return response code 3009.
Response Code	3009
Response Description	The transaction number identified from registry already exists in the ITL. This message cannot be processed.

6. Message Sequence Checks for Transactions from STL

Check Name	Transaction Status Not Compatible with STL
Check Description	Transaction status <> "STL Checked (No Discrepancy)" or "STL Checked (Discrepancy)".
Process	If transaction status returned from the STL <> 9 or 10, return response code 3501.
Response Code	3501
Response Description	Invalid transaction status from STL. This message cannot be processed.

Check Name	Transaction ID Presence
Check Description	Transaction ID must exist in ITL.
Process	If the input transaction status code > 1, Select Transaction_ID From Transaction_Log Where Transaction_ID = input transaction ID If not found, return response code 3502.
Response Code	3502
Response Description	Transaction ID does not exist in ITL. This message cannot be processed.

7. General Checks for Transactions

Check Name	Units are Unavailable
Check Description	The units identified in the transaction must be available (i.e., not involved in another transaction).
Process	For every block in the ITLUnitBlockObject array, If Unit_Status_Code = 1, 3, 5, or 7, return response code 4001.
Response Code	4001
Response Description	Transaction units are part of an ongoing transaction. This message cannot be processed.

Check Name	Units are Cancelled
Check Description	Cancelled units cannot be traded.
Process	For each unit block in the ITLUnitBlockObject If the Account_Type_Code = 210, 220, 230 or 240, return response code 4002.
Response Code	4002
Response Description	Transaction units have been previously cancelled. This message cannot be processed.

Check Name	Units are Retired
Check Description	Retired units cannot be traded.
Process	For each unit block in the ITL UnitBlockObject array, If the Account_Type_Code = 300, return response code 4003.
Response Code	4003
Response Description	Transaction units are retired. This message cannot be processed.

Check Name	Units Have Inconsistencies
Check Description	Units identified in the transaction cannot have existing reconciliation inconsistencies.
Process	For each unit block in the ITLUnitBlockObject array, if Unit_Status_Code = 2, 3, 6, 7, return response code 4004.
Response Code	4004
Response Description	Transaction units have been flagged for inconsistency. This message cannot be processed.

Check Name	Registry Holds Units
Check Description	Units identified in transaction must be held by Initiating Registry.
Process	For each unit block in the ITLUnitBlockObject array, if Holding_Registry_Code <> input Initiating Registry, return response code 4005.
Response Code	4005
Response Description	Units are not found in Initiating Registry. This message cannot be processed.

Check Name	Current Commitment Period
Check Description	Every unit in a proposed transaction must have an applicable Commitment Period identifier consistent with the current Commitment Period (and its grace period) or for the next Commitment Period.
Process	For each unit block in the transaction, If the applicable Commitment Period is less than the current Commitment Period identified in the system parameter table, return response code 4006.
Response Code	4006
Response Description	The applicable Commitment Period for unit blocks in this transaction is no longer valid. This message cannot be processed.

8. Transaction-specific Checks

Check Name	AAU Issued Amount Check
Transaction Type(s)	Issuance
Check Description	The number of AAUs issued must not exceed allocation.
Process	Determine the total number of units to be issued by type and commitment period. For each unit type and commitment period: 1. Determine previously issued units for type and commitment period. Select Sum(c.End_Block – c.Start_Block + 1) as PreviouslyIssued From Transaction_Log a, inner join Transaction_Status_History b, inner join Transaction_Block c Where a.transaction_type = 1 and b.transaction_status_datetime = most recent status and b.transaction_status = 4 and c.unit_type = 1 and c.originating_commitment_period = input commitment period 2. Determine number allowed for type and commitment period. Select Value From Registry_Attributes where attribute type code = 1 (Total Allocation AAUs) and Registry_Code = input registry code and Period_Code = input commitment_period 3. If Units to be Issued + PreviouslyIssued > Value, return response code 5001.
Response Code	5001
Response Description	AAU amount issued exceeds the amount available for issuance.

Check Name	RMU Issued Amount
Check Description	The number of RMUs issued must not exceed allowance for activity type.
Process	Determine the total number of units to be issued by type and commitment period. For each unit type and commitment period: 1. Determine previously issued RMUs by activity type and commitment period. Select Sum(c.End_Block – c.Start_Block + 1) as PreviouslyIssued From Transaction_Log a, inner join Transaction_Status_History b, inner join Transaction_Block c Where a.transaction_type = 1 and b.transaction_status_datetime = most recent status and b.transaction_status = 4 and c.unit_type = 2 and c.originating_commitment_period = input commitment period and c.LULUCF code = [1, 2, or 3] 2. Determine number allowed for type and commitment period. Select Value From Registry_Attributes where attribute_type_code = 5,6, or 7 (Total Issuable RMUs for Activity Types [1, 2, or 3]) and Registry_Code = input registry code and Period_Code = input commitment_period 3. If Units to be Issued + PreviouslyIssued > Value, return response code 5002.
Response Code	5002
Response Description	RMU amount issued exceeds the amount available for issuance for this activity type.

877
878

Check Name	RMU Issuance
Transaction Type(s)	Issuance
Check Description	RMUs cannot be issued before the end of the Commitment Period unless annual issuance option is selected for this LULUCF activity.
Process	If unit type code = 2, 1. Determine Current Commitment Period Select [Commitment Period] From System_Paramter Where [Commitment Period] = [current Commitment Period] 2. Check Method of RMU Issuance for this registry Select Value From Registry_Attribute Where Attribute_Type_Code = 4 and Registry_Code = Initiating Registry and Period_Code = current Commitment Period 3. If found and current date < current Commitment Period +1 , return response code 5003.
Response Code	5003
Response Description	RMUs issued before end of the Commitment Period. This message cannot be processed.

Check Name	Consistency of Unit Type Issued for a Project
Transaction Type(s)	Issuance
Check Description	Choice of issuing tCERs or lCERS must be consistent with previous issuances for the project.
Process	If Unit_Type to be issued = (6 or 7), Select Unit_Type_code, From Project Where Project_ID = input Project ID If Unit_Type <> issued unit type, return response code 5004.
Response Code	5004
Response Description	Unit type proposed to be issued is inconsistent with the unit type previously issued for this Project. This message cannot be processed.

Check Name	Issuing tCERs or lCERs for Project Type
Transaction Type(s)	Issuance
Check Description	tCERs and lCERs must be issued for a forestation project.
Process	If Unit_Type to be issued = (6 or 7), and LULUCF code of issued units <> 1 return response code 5005.
Response Code	5005
Response Description	The unit type issued is not related to a forestation project. This message cannot be processed.

883
884

Check Name	Expiry Date Check
Transaction Type(s)	Issuance
Check Description	tCERs and lCERs must have expiry date.
Process	If unit type code = (6, 7) and expiry date is null or blank, return response code 5006.
Response Code	5006
Response Description	The tCERs or lCERs do not have an expiry date. This message cannot be processed.

885
886

Check Name	tCER Expiry Date
Transaction Type(s)	Issuance
Check Description	Expiry date for tCERs must be end of second Commitment Period.
Process	If unit type code = 6 and expiry date is > date of second Commitment Period, return response code 5007.
Response Code	5007
Response Description	The tCER Expiry date is invalid. This message cannot be processed.

887
888

Check Name	lCER Expiry Date
Transaction Type(s)	Issuance
Check Description	Expiry date for lCERs must be consistent with Project crediting period dates.
Process	If unit type code = 7, Select Crediting_Period_End_Date From Project Where Project_ID = unit project ID. If unit Expiry date <> Crediting_Period_End_Date, return response code 5008.
Response Code	5008
Response Description	The lCER expiry date is inconsistent with Project dates approved by CDM Executive Board. This message cannot be processed.

Check Name	CDM Issuance of lCERs, tCERs and CERs
Transaction Type(s)	Issuance
Check Description	Only the CDM Registry may issue CERs, lCERs and tCERs.
Process	If unit type code = 5, 6, 7, If "from" or Initiating Registry <> CDM, return response code 5009.
Response Code	5009
Response Description	A national registry is not allowed to issue CERs, tCERs or lCERs. This message cannot be processed.

Check Name	CDM Issuance of Other Unit Types
Transaction Type(s)	Issuance
Check Description	CDM Registry cannot issue unit types other than CERs, lCERs or tCERs.
Process	If Initiating Registry = CDM, and unit type code <> (5, 6, 7), else return response code 5010.
Response Code	5010
Response Description	The CDM Registry cannot issue this unit type. This message cannot be processed.

Check Name	ERU Conversion Quantity
Transaction Type(s)	Conversion
Check Description	ERU Conversion for each Track 2 JI Project must not exceed quantity specified by the Article 6 Supervisory Committee.
Process	If unit type code = 3, select Project_Status_Code from Project where Project_ID = unit Project ID and Track = 2 If not found or if found and Project_Status_Code <> 2, return response code 5011.
Response Code	5107
Response Description	The quantity of ERU's converted exceeds the quantity specified by the Article 6 Supervisory Committee.

Check Name	CER Issuance Amount for Project
Transaction Type(s)	Issuance
Check Description	CER Issuance for each CDM Project by CDM Registry must not exceed amount specified by CDM Executive Board.
Process	If unit type code = 5, 1. Determine the number of units to be issued in this transaction. 2. Determine the total number of units already issued for this Project: Select Sum(c.End_Block - c.Start_Block + 1) as PreviouslyIssued From Unit_Block Where Unit_Type = 5 and Project_ID = input Project ID 3. Determine number of units allowed to be issued for this project. Select Total_Units as AllowedUnits From Project_Action_Log Where Project_ID = input Project ID and Project_Action_Code = 1 and Action_Date = most recent status for this Project where Project_Action_Code = 1 4. If Units to be Issued + PreviouslyIssued > AllowedUnits, return response code 5012.
Response Code	5012
Response Description	CER Issuance for each CDM Project by CDM Registry must not exceed amount specified by CDM Executive Board. This message cannot be processed.

898
899

Check Name	tCER Issuance Amount for Project
Transaction Type(s)	Issuance
Check Description	tCER Issuance for each CDM Project by CDM Registry must not exceed amount specified by CDM Executive Board.
Process	If unit type code = 6, 1. Determine the number of units to be issued in this transaction. 2. Determine the total number of units already issued for this project: Select Sum(c.End_Block - c.Start_Block + 1) as PreviouslyIssued From unit_block Where unit_type = 6 and Project_ID = input project id 3. Determine number of units allowed to be issued for this project. Select Total_Units as AllowedUnits From Project_Action_Log Where Project_ID = input project id and Project_Action_Code = 1 and action_date = most recent status for this project where Project_Action_Code = 1 4. If Units to be Issued + PreviouslyIssued > AllowedUnits, return response code 5013.
Response Code	5013
Response Description	tCER issuance for each CDM Project by CDM Registry exceeds approved number of units for this Project. This message cannot be processed.

900
901

Check Name	lCER Issuance Amount for Project
Transaction Type(s)	Issuance
Check Description	lCER Issuance for each CDM Project by CDM Registry must not exceed amount specified by CDM Executive Board.
Process	If unit type code = 7, 1. Determine the number of units to be issued in this transaction. 2. Determine the total number of units already issued for this Project: Select Sum(End_Block - Start_Block + 1) as PreviouslyIssued From unit_block Where unit_type = 7 and Project_ID = input project id 3. Determine number of units allowed to be issued for this Project. Select Total_Units as AllowedUnits From Project_Action_Log Where Project_ID = input project id and Project_Action_Code = 1 and Action_Date = most recent status for this project where Project_Action_Code = 1 4. If Units to be Issued + PreviouslyIssued > AllowedUnits, return response code 5014.
Response Code	5014
Response Description	lCER issuance for each CDM Project by CDM Registry exceeds approved number of units for this Project. This message cannot be processed.

902
903

Check Name	Issued Serial Numbers
Check Description	The serial numbers issued must be unique.
Process	For each unit block to be issued for transaction type 1 (or 10-51, 52, 54), Select Block_ID From Unit_Block Where Originating_Registry_Code = input Originating Registry and ((Start_Block > input start block and Start_Block < input end block) or (End_Block > input start block and End_Block < input end block) or (Start_Block > input start block and End_Block < input end block)) If found, return response code 5015.
Response Code	5015
Response Description	Proposed serial numbers identified are not unique. This message cannot be processed.

904
905

Check Name	Expired tCERs and lCERs Trade Restrictions
Transaction Type(s)	Conversion, External, Retirement, Replacement, Carry-over
Check Description	If a tCER or lCER has expired, it may not be involved in any transaction, except in an internal transfer to a Type 1 voluntary cancellation account.
Process	If unit type = (6 or 7) and transaction type <> 1 or 4 Evaluate the expiry date in the ITLUnitBlockObject. If Expiry_Date ≤ Current date, return response code 5030.
Response Code	5030
Response Description	One or more of the units identified in this serial block have expired lCERs or tCERs. Expired tCERs and lCERs may not be traded or modified, except to be transferred to a cancellation account. This message cannot be processed.

906
907

Check Name	Replacement Unit Trade Restriction
Transaction Type(s)	Conversion, External, Cancellation, Retirement, Replacement, Carry-over, Expiry Date Change
Check Description	A unit previously designated to replace another unit may not be involved in any other transaction.
Process	Using the Block ID from the ITLUnitBlockObject, search Replacement_Unit_Block table for prior replacement. Select From Replacement_Unit_Block Where Block_ID = Block_ID of units to be transferred If found, return response code 5031.
Response Code	5031
Response Description	One or more of the units identified in this serial block have already been designated as replacement units. Replacement units may not be traded or modified. This message cannot be processed.

908
909

Check Name	Flagged Unit Check
Transaction Type(s)	Conversion, External, Cancellation, Retirement, Replacement, Carry-over, Expiry Date Change
Check Description	If the associated Project has not submitted a certification report or had a reversal of storage, lCERs generated from the Project may only be transferred to Type 2 or Type 3 cancellation accounts.
Process	If unit_type = 7 and if Project Freeze Flag of ITLUnitBlockObject = 1 if transaction type = 2, 3, 5, 6, 7, or 8 or if transaction type = 4 and and acquiring account type <> 422 or 423 return response code 5032.
Response Code	5032
Response Description	One or more of the units identified in this serial block are lCERs from a Project that has had a reversal of greenhouse gas net removal by the sink or from a Project that has not provided a timely certification report. This message cannot be processed.

910

Check Name	Unit Block Attributes
Transaction Type(s)	Retirement, Cancellation, Replacement and External
Check Description	All attributes of a unit block must be consistent with ITL unit block attributes for Retirement, Cancellation, Replacement and External transactions.
Process	For each unit block identified in a transaction with the transaction type of 3, 4, 5 or 6: Select Block_ID From Unit_Block Where Originating_Registry_Code = input Originating Registry and Start_Block ≤ input start block and End_Block ≥ input end block. If input Original Commitment Period <> Original Commitment Period or If input Applicable Commitment Period <> Applicable Commitment Period or If input LULUCF code <> LULUCF Code or If input Project Identifier <> Project Identifier or If input Track <> Track or If Expiry date <> Expiry date, return response code 5033
Response Code	5033
Response Description	The attributes of the unit block are inconsistent with the ITL unit block attributes.

911
912

Check Name	Units Available for Carry-over
Transaction Type(s)	Carry-over
Check Description	Units identified in serial blocks must be specified as being available to be carried-over.
Process	Select From C&A Approved Units Table [future] Where Originating_Registry_code = unit Originating Registry code and Start_Serial <= unit serial start and End_Serial >= unit serial end If not found, return response code 5050.
Response Code	5050
Response Description	One or more of the units identified in this serial block have not been specified to be carried over. All units to be carried over must be listed in the registry's final compilation and accounting report. This message cannot be processed.

913

914

Check Name	RMU Carry-over
Transaction Type(s)	Carry-over
Check Description	RMUs may not be carried-over.
Process	For each unit block in the ITLUnitBlockObject, If unit type = 2, return response code 5051.
Response Code	5051
Response Description	One or more of the units identified in this serial block is an RMU which cannot be carried-over. This message cannot be processed.

915
916

Check Name	ERU Carry-over
Transaction Type(s)	Carry-over
Check Description	ERUs converted from RMUs may not be carried-over.
Process	For each unit block in the ITLUnitBlockObject, If unit type = 4, return response code 5052.
Response Code	5052
Response Description	One or more of the units identified in this serial block is an ERU that originated as an RMU. These units cannot be carried over. This message cannot be processed.

917
918

Check Name	tCER or lCER Carry-over
Transaction Type(s)	Carry-over
Check Description	tCERs or lCERs may not be carried over.
Process	For each unit block in the ITLUnitBlockObject, If unit type = (6 or 7), return response code 5053.
Response Code	5053
Response Description	One or more of the units identified in this serial block is a tCER or lCER which cannot be carried over. This message cannot be processed.

919
920

Check Name	ERU Carry-over Limit
Transaction Type(s)	Carry-over
Check Description	Total quantity of ERUs for Carry-over must not exceed limit.
Process	If unit type code = 3 1. Determine number of units involved in transaction 2. Determine number of ERUs already carried-over Select Sum(End_Block - Start_Block + 1) as PreviouslyCarriedOver From unit_block Where unit_type = 3 or 4 and Applicable_Commitment_period_Code = Original_Commitment_Period_Code + 1 and Applicable_Commitment_Period_Code = Current Commitment Period 3. Determine number of ERUs allowed for Carry-over Select Value From Registry_Attribute Where Registry_Code = initiating registry code and Attribute_Type_Code = 1 and Period_Code = originating period code AllowedNumber = 0.025 * Value 4. If number of ERUs to be carried-over + number of ERUs already carried-over > Allowed Number, return response_code 5054
Response Code	5054
Response Description	Total number of ERUs to be carried over exceeds limit. This message cannot be processed.

921
922

Check Name	CER Carry-over Limit
Transaction Type(s)	Carry-over
Check Description	Total quantity of CERs to be carried over must not exceed limit.
Process	If unit type code = 5 1. Determine number of units involved in transaction 2. Determine number of CERs already carried-over Select Sum(End_Block - Start_Block + 1) as PreviouslyCarriedOver From unit_block Where unit_type = 5 and Applicable_Commitment_Period_Code = Original_Commitment_Period_Code + 1 and Applicable_Commitment_Period_Code = current Commitment Period 3. Determine number of CERs allowed for Carry-over Select Value From Registry_Attribute Where registry_code = Initiating Registry code and Attribute_type_code = 1 and Period_Code = originating commitment period code AllowedNumber = 0.025 * Value 4. If number of CERs to be carried-over + number of CERs already carried-over > Allowed Number, return response_code 5055
Response Code	5055
Response Description	Total number of CERs to be carried over exceeds limit. This message cannot be processed.

923
924

Check Name	Unit Block Attributes for Carry-over Transactions
Transaction Type(s)	Carry-over
Check Description	All attributes except for Applicable Commitment Period of a unit block must be consistent with ITL unit block attributes for Carry-over transactions.
Process	For each unit block identified in a transaction with the transaction type of 7: Select Block_ID From Unit_Block Where Originating_Registry_Code = input Originating Registry and Start_Block ≤ input start block and End_Block ≥ input end block. If input original Commitment Period <> original Commitment Period or If input LULUCF code <> LULUCF code or If input Project Identifier <> Project Identifier or If input track <> track or If expiry date <> expiry date, return response code 5056.
Response Code	5056
Response Description	The attributes of the unit block are inconsistent with the ITL unit block attributes.

925
926

Check Name	Conversion Eligibility (Track 1)
Transaction Type(s)	Conversion
Check Description	If the unit is a Track 1 ERU, then the Party must meet all 6 eligibility criteria.
Process	If the unit track = 1, for i = 1 to 6 Select Registry_Code From Registry_Eligibility Where Registry_Code = Initiating Registry code and Criteria_End_Date is null and Criteria_Type_Code = 1 and Eligibility_Status_Code = 1 If not found, stop processing and return response code 5101.
Response Code	5101
Response Description	Party is not eligible for Track 1 conversions. This message cannot be processed.

927
928

Check Name	Conversion Eligibility (Track 2)
Transaction Type(s)	Conversion
Check Description	If the unit is a Track 2 ERU then the Party must meet eligibility criteria 1, 2 and 4.
Process	If the unit track = 2, Execute the following query 3 times, once each for criteria type code = 1, 2 and 4 Select Registry_Code From Registry_Eligibility Where Criteria_End_Date is null and Registry_Code = Initiating Registry code If one or more not found, return response code 5102.
Response Code	5102
Response Description	Party is not eligible for Track 2 conversions. This message cannot be processed.

929
930

Check Name	Conversion Unit Type
Transaction Type(s)	Conversion
Check Description	Units for conversion must be AAUs or RMUs
Process	For each unit block in the ITL UnitBlockObject, If unit type <> (1 or 2), return response code 5103.
Response Code	5103
Response Description	Units for conversion are not AAUs or RMUs. This message cannot be processed.

931
932

Check Name	Initiating Registry for Conversion
Transaction Type(s)	Conversion
Check Description	Units for conversion must be issued by Initiating Registry.
Process	For each unit block in the ITL UnitBlockObject, If Originating Registry code <> Transferring Registry code, return response code 5104.
Response Code	5104
Response Description	Units for conversion were not issued by initiating Registry. This message cannot be processed.

933
934

Check Name	Proposing Registry
Transaction Type(s)	Conversion
Check Description	The Initiating Registry converting AAUs or RMUs must be a national registry.
Process	If the Initiating Registry code = CDM, ITL, STL, return response code 5105
Response Code	5105
Response Description	Only national registries can initiate conversion transactions.

935
936

Check Name	Unit Block Attributes for Conversion
Transaction Type(s)	Conversion
Check Description	All attributes of a unit block, except for the Project ID and the unit type, must be consistent with ITL unit block attributes for conversion transactions.
Process	For each unit block identified in a transaction with the transaction type of 2, Select Block_ID From Unit_Block Where Originating_Registry_Code = input Originating Registry and Start_Block ≤ input start block and End_Block ≥ input end block. If input Original Commitment Period <> Original Commitment Period or If input Applicable Commitment Period <> Applicable Commitment Period or If Expiry date <> Expiry date return response code 5106
Response Code	5106
Response Description	The attributes of the unit block are inconsistent with the ITL unit block attributes.

Check Name	CER, tCER and lCER Retirement Limit
Transaction Type(s)	Retirement
Check Description	Total quantity of CERs, tCERs and lCERs retired must not exceed limit.
Process	If unit type code = 5, 6, 7 1. Determine number of units involved in transaction 2. Determine number of CERs, tCERs, and lCERs already retired Select Sum(End_Block - Start_Block + 1) as PreviouslyRetired From unit_block Where unit_type = 5, 6, 7 and Applicable_Commitment_Period_Code = current Commitment Period and Holding_Registry_Code = Initiating Registry and Account_Type_Code = 300 3. Determine number of units allowed for retirement Select Value From Registry_Attribute Where registry_code = Initiating Registry code and Attribute_type_code = 1 and Period_Code = originating period code AllowedNumber = 0.5 * Value 4. If number of CERs, tCERs, and lCERs to be carried-over + number already carried-over > Allowed Number, return response_code 5150
Response Code	5150
Response Description	The number of CERs, tCERs or lCERs to be retired exceeds the retirement limit for lCERs and tCERs. The limit is 5% of the number of AAUs allowed to be issued for the Commitment Period. This message cannot be processed.

945
946

Check Name	CER, tCER and lCER Retirement Eligibility
Transaction Type(s)	Retirement
Check Description	To retire CERs, tCERs or lCERs, a Party must meet all 6 eligibility criteria.
Process	If the unit track = 1, for i = 1 to 6 Select Registry_Code From Registry_Eligibility Where Registry_Code = Initiating Registry code and Criteria_End_Date is null and Criteria_Type_Code = 1 and Eligibility_Status_Code = 1 If not found, stop processing and return response code 5151.
Response Code	5151
Response Description	Party is not eligible for retirements of CERs, tCERs or lCERs. This message cannot be processed.

947
948

Check Name	External Transfers to CDM
Transaction Type(s)	External
Check Description	CDM Registry can only receive transfers to Cancellation accounts.
Process	If the Acquiring Registry = CDM and the acquiring Account_Type code does not equal 210, 220, 230 or 240 return response code 5200.
Response Code	5200
Response Description	CDM Registry can only receive transfers to cancellation accounts. This message cannot be processed.

949
950

Check Name	Acquiring Registry Eligibility for External Transfers
Transaction Type(s)	External
Check Description	The Acquiring Registry must meet all six eligibility criteria.
Process	i = 1 – 6 Select Registry_Code From Registry_Eligibility Where Registry_Code = Acquiring Registry code and Criteria_End_Date is null and Criteria_Type_Code = i and Eligibility_Status_Code = 1 If not found, stop processing and return response code 5202.
Response Code	5202
Response Description	The Acquiring Registry is not eligible to participate in external transfers. This message cannot be processed.

951
952

Check Name	Commitment Period Reserve Level
Transaction Type(s)	External
Check Description	The proposed External transfer must not result in the total registry holdings of all units to fall below the CPR level for the transferring Party.
Process	1. Determine number of units involved in transaction 2. Determine number of units held by registry Select Sum(End_Block - Start_Block + 1) as TotalHoldings From unit_block and applicable_period_code = current Commitment Period and Holding_Registry_Code = Initiating Registry 3. Determine Commitment Period Reserve Select Value as CPR From Registry_Attribute Where registry_code = Initiating Registry code and attribute_type_code = 2 and period_code = originating period code 4. If Registry Holdings - Units involved in transaction < CPR, return response_code 5211
Response Code	5211
Response Description	The proposed transaction will cause the registry holdings to fall below the CPR level. This message cannot be processed.

953
954

Check Name	Number of Replacement Units
Transaction Type(s)	Replacement
Check Description	The number of units replaced must equal the number of replacing units.
Process	1. Determine total number of units to be replaced by this transaction. 2. Determine total number of units traded by this transaction. If Step 1 <> Step 2, return response code 5301.
Response Code	5301
Response Description	In a replacement transaction, the number of units replaced must equal the number of replacing units. This message cannot be processed.

955
956

Check Name	One-To-Many Replacement Blocks
Transaction Type(s)	Replacement
Check Description	A transaction may not contain many-to-many relationships between replaced and replacing blocks.
Process	Determine number of blocks to be replaced. Determine number of replacing blocks. If both > 1, return response code 5302.
Response Code	5302
Response Description	The transaction includes multiple replaced blocks and multiple replacement blocks. This message cannot be processed.

957
958

Check Name	Location of Replaced tCERs
Transaction Type(s)	Replacement
Check Description	Only tCERs held in a Retirement account or Replacement account may be replaced.
Process	Evaluate current holding account type in the ITLUnitBlockObject. if Account_Type_Code = (300, 411), return response code 5303.
Response Code	5303
Response Description	tCERs to be replaced are not held in a retirement account or a replacement account. This message cannot be processed.

959
960

Check Name	Replacement Registry
Transaction Type(s)	Replacement
Check Description	The registry holding the units to be replaced and the replacing units must be the same.
Process	If units to be replaced registry code <> Acquiring Registry code, return response code 5304.
Response Code	5304
Response Description	The holding registry for the unit to be replaced is not the same as the registry for the replacing units. This message cannot be processed.

961
962

Check Name	Replacement Account Type
Transaction Type(s)	Replacement
Check Description	Only tCER Replacement accounts and lCER Replacement accounts may acquire lCERs in a replacement transaction.
Process	If acquiring account type code <> 411, 421, 431, 441 return response code 5305.
Response Code	5305
Response Description	The acquiring account type is not a tCER or lCER Replacement account. This message cannot be processed.

963
964

Check Name	Location of Replaced lCERs
Transaction Type(s)	Replacement
Check Description	lCERs that are held in Cancellation accounts may not be replaced.
Process	If the account type for the lCERs <> 421, 422, 423 return response code 5306.
Response Code	5306
Response Description	Only lCERs that are held in cancellation accounts may be replaced.

965
966

Check Name	lCER Replacement Due to Expiry
Transaction Type(s)	Replacement
Check Description	lCER Replacement accounts (unit expiry) may not acquire tCERs or lCERs.
Process	If acquiring account type = 421, If replacing unit_type must <> (1, 2, 3, 4, 5 or 6), return response code 5308.
Response Code	5308
Response Description	For a replacement transaction due to lCER expiry, the only allowed unit types are AAUs, RMUs, CERs, and ERUs. This message cannot be processed.

967
968

Check Name	lCER Replacement Units (Due to Lack of Certification Report or Reversal of Storage)
Transaction Type(s)	Replacement
Check Description	lCER Replacement accounts (lack of certification or Reversal in Storage) may not acquire tCERs and may not acquire lCERs with a Project Identifier other than that specified in the replacement notification.
Process	If acquiring account type = 422, 423 If NOT (replacing Unit_Type = (1, 2, 3, 4 or 5), or (replacing Unit_Type = 7 and replacing project ID = project ID of replaced units)) return response code 5309.
Response Code	5309
Response Description	lCER Replacement accounts (lack of certification or Reversal in Storage) may not acquire tCERs and may not acquire lCERs with a Project Identifier other than that specified in the replacement notification. This message cannot be processed.

969
970

Check Name	Multiple Replacement
Transaction Type(s)	Replacement
Check Description	A unit may be replaced only once.
Process	Using the Block ID from the ITLUnitBlockOject, search Replacement_Unit_Block table for prior replacements. Select all From Replacement_Unit_Block Where Block_ID = Block_ID of units to be replaced If found, return response code 5310.
Response Code	5310
Response Description	One or more of the units designated to be replaced have already been replaced by other units. This message cannot be processed.

971
972

Check Name	tCER Replacement Units (Due to Expiry)
Transaction Type(s)	Replacement
Check Description	tCER replacement accounts (unit expiry) may not acquire lCERs.
Process	If replacement transaction due to expiry, If acquiring account type = 410, If replacing unit_type <> (1, 2, 3, 4, 5 or 6), return response code 5311.
Response Code	5311
Response Description	If a replacement transaction due to tCER expiry, the only allowed unit types are AAUs, RMUs, ERUs, CERs, and tCERs. This message cannot be processed.

973
974

Check Name	tCER or lCER Cancellation Purposes
Transaction Type(s)	Cancellation
Check Description	tCERs and lCERs may not be transferred to Type 1, 2 or 4 Cancellation accounts.
Process	If the acquiring account type = 210, 220, 240 return response code 5401.
Response Code	5401
Response Description	tCERs and lCERs may not be transferred to Type 1, 2, or 4 Cancellation accounts. This message cannot be processed.

975
976

Check Name	Units for Expiry Date Change
Transaction Type(s)	Expiry Date Change
Check Description	The units for Expiry Date Change must be tCERs or lCERs.
Process	If unit type code <> 6 or 7, return response code 5450.
Response Code	5450
Response Description	The units for expiry date change are not tCERs or lCERs. This message cannot be processed.

977
978

Check Name	New tCER Expiry Date
Transaction Type(s)	Expiry Date Change
Check Description	The new tCER expiry date must be consistent with the end date of the second Commitment Period.
Process	If unit type code = 6, Query System_Parameter table Select [Expiry date] From System_Paramter Where [Commitment Period] = unit Original Commitment Period + 1 If Commitment Period Expiry date <> unit expiry date, return response code 5451.
Response Code	5451
Response Description	The new tCER expiry date is not consistent with the end date of the second Commitment Period. This message cannot be processed.

Check Name	New lCER Expiry Date
Transaction Type(s)	Expiry Date Change
Check Description	The new lCER expiry date must be consistent with the end date of the renewed crediting period for the project.
Process	If unit type code = 7, Select crediting_period_end_date From project where project_ID = unit project ID and crediting_period_end_date = unit expiry date If not found, return response code 5452.
Response Code	5452
Response Description	The new lCER expiry date must be consistent with the end date of the renewed crediting period for the Project.

Check Name	Unit Block Attributes for Expiry Date Change Transactions
Transaction Type(s)	Expiry Date Change
Check Description	All attributes of a unit block except for the Expiry date must be consistent with ITL unit block attributes for Expiry Date Change transactions.
Process	For each unit block identified in a transaction with the transaction type of 7: Select Block_ID From Unit_Block Where Originating_Registry_Code = input Originating Registry and Start_Block ≤ input start block and End_Block ≥ input end block. If input original Commitment Period <> original Commitment Period or If input applicable Commitment Period <> applicable Commitment Period or If input LULUCF code <> LULUCF code or If input Project Identifier <> Project Identifier or If input track <> track return response code 5453
Response Code	5453
Response Description	The attributes of the unit block are inconsistent with the ITL unit block attributes.

9. Data Integrity Checks for Reconciliation

Check Name	Reconciliation Identifier
Check Description	Reconciliation Identifier must be greater than zero.
Process	If input reconciliation identifier is ≤ zero, return response code 6201.
Response Code	6201
Response Description	Reconciliation identifier must be > zero. This message cannot be processed.

Check Name	Reconciliation Mask
Check Description	Reconciliation ID must be comprised of a valid registry code followed by numeric values.
Process	Determine if first 2 or 3 characters are alpha. Determine if remaining characters are numeric. return response code 6202.
Response Code	6202
Response Description	Reconciliation ID has invalid format. This message cannot be processed.

989
990

Check Name	Reconciliation Status Validity
Check Description	Reconciliation status must be a value between 1 and 11.
Process	If input reconciliation status code < 1 or >11, return response code 6203.
Response Code	6203
Response Description	Reconciliation status code invalid. This message cannot be processed.

991
992

Check Name	Reconciliation Snapshot DateTime
Check Description	Reconciliation snapshot must be a date between 01-OCT-2004 and the current date plus 30 days.
Process	If input reconciliation snapshot date < 01-OCT-2004 or input reconciliation snapshot date > current date + 30 days, return response code 6204.
Response Code	6204
Response Description	Reconciliation snapshot date is invalid. This message cannot be processed.

993
994

Check Name	Account Type Validity
Check Description	Account type must be valid.
Process	If input account type not in Account_Type_Code table, return response code 6205.
Response Code	6205
Response Description	Account type is invalid. This message cannot be processed.

995
996

Check Name	Unit Type Validity
Check Description	Unit type must be valid.
Process	If input unit type not in Unit_Type_Code table, return response code 6206.
Response Code	6206
Response Description	Unit type is invalid. This message cannot be processed.

Check Name	Supplementary Unit Type Validity
Check Description	Supplementary unit type must be valid.
Process	If input supplementary unit type not in Supp_Unit_Type code table, return response code 6207.
Response Code	6207
Response Description	The supplementary unit type is invalid. This message cannot be processed.

Check Name	Reconciliation Phase
Check Description	Reconciliation Phase must be valid.
Process	If input reconciliation phase code not in Reconciliation_Phase_Code table, return response code 6208.
Response Code	6208
Response Description	The Reconciliation Phase code is invalid. This message cannot be processed.

10. Reconciliation Message Sequence for Registry Messages

Check Name	Reconciliation ID Does Not Exist
Check Description	Reconciliation ID must exist in the Reconciliation Log table.
Process	Select Reconciliation_ID From Reconciliation_Log Where Reconciliation_ID = input reconciliation identifier If not found, return response code 6301.
Response Code	6301
Response Description	Invalid reconciliation identifier sent by registry.

Check Name	Reconciliation Status Not Valid
Check Description	Out of Sequence reconciliation status sent by registry is invalid.
Process	If input reconciliation status = (2, 5, 6, 7, 10, 11), return response code 6302.
Response Code	6302
Response Description	The reconciliation status sent by the registry is invalid.

Check Name	Reconciliation Status Out of Sequence
Check Description	Incoming reconciliation status should be the same as the reconciliation status recorded by the ITL.
Process	Select Recon_Status_Code From Reconciliation_Status_History Where Recon_Log_DateTime = most recent date for this recon ID and Recon_ID = input recon ID and Recon_Status_Code = input recon status code. If not found, return response code 6303.
Response Code	6303
Response Description	Invalid reconciliation identifier sent by registry.

Check Name	Reconciliation Snapshot DateTime
Check Description	The registry reconciliation snapshot datetime must be consistent with the ITL Reconciliation Snaphot DateTime.
Process	Verify that snapshot DateTime for reconciliation ID matches the ITL reconciliation log table, else return response code 6304.
Response Code	6304
Response Description	The snapshot date and time provided does not match with requested snapshot date and time. This message cannot be processed.

11. Reconciliation Sequence Checks from STL

Check Name	Reconciliation ID Sent by STL Does Not Exist
Check Description	Reconciliation ID sent by the STL must already exist in the ITL unless the STL is requesting the ITL to initiate a new reconciliation action.
Process	If input reconciliation status <> 1 (Initiated), Select Recon_ID From Reconciliation_Log Where Recon_ID = input reconciliation ID is not found, or If input reconciliation status = 1 and the input Recon_ID <> null, return response code 6311.
Response Code	6311
Response Description	Invalid reconciliation identifier sent by STL.

Check Name	Reconciliation Status Not Valid
Check Description	Reconciliation status sent by the STL must be one of certain enumerated statuses.
Process	If input reconciliation status code is: 2 (Validated) 3 (Totals Inconsistent) 4 (Unit Blocks Inconsistent) 5 (Completed) 6 (Completed with Manual Intervention) 7 (Start Request Denied) return response code 6312.
Response Code	6312
Response Description	Out of sequence reconciliation status sent by STL.

Check Name	Reconciliation Status of "STL Totals Inconsistent" is Out of Sequence
Check Description	If the incoming reconciliation status is "STL Totals Inconsistent," the previously recorded status at the ITL must be "Validated."
Process	If input reconciliation status code is (8), Select Recon_Log_ID From Reconciliation_Status_History Where Recon_ID = input reconciliation ID and Recon_Log_DateTime = most recent reconciliation status and Recon_Status_Code = "2" If not found, return response code 6313.
Response Code	6313
Response Description	Reconciliation status of "STL Totals Inconsistent" is out of sequence with the reconciliation status recorded in the ITL.

1020
1021

Check Name	Reconciliation Status of "STL Unit Blocks Inconsistent" Out of Sequence
Check Description	If the incoming reconciliation status is "STL Unit Blocks Inconsistent", the previously recorded status at the ITL must be "STL Totals Inconsistent."
Process	If input reconciliation status code is (9), Select Recon_Log_ID From Reconciliation_Status_History Where Recon_ID = input reconciliation ID and Recon_Log_DateTime = most recent reconciliation status and Recon_Status_Code = "8" If not found, return response code 6314.
Response Code	6314
Response Description	Reconciliation status of "STL Unit Blocks Inconsistent" is out of sequence with the reconciliation status recorded in the ITL.

1022
1023

Check Name	Reconciliation status of "STL Validated" is out of sequence.
Check Description	If the incoming reconciliation status is "STL Validated," the previously recorded status at the ITL must be "Validated," "STL Totals Inconsistent," or "STL Unit Blocks Inconsistent."
Process	If input reconciliation status code is (10), Select Recon_Log_ID From Reconciliation_Status_History Where Recon_ID = input reconciliation ID and Recon_Log_DateTime = most recent reconciliation status and Recon_Status_Code = (2, 8 or 9) If not found, return response code 6315.
Response Code	6315
Response Description	Reconciliation status of "STL Validated" is out of sequence with the reconciliation status recorded in the ITL.

1024
1025

Check Name	Reconciliation Status of "STL Complete with Manual Intervention" is Out of Sequence
Check Description	If the incoming reconciliation status is "STL Complete with Manual Intervention," the previously recorded status at the ITL must be "STL Totals Inconsisten," or "STL Unit Blocks Inconsistent."
Process	If input reconciliation status code is (11), Select Recon_Log_ID From Reconciliation_Status_History Where Recon_ID = input reconciliation id and Recon_Log_DateTime = most recent reconciliation status and Recon_Status_Code = (8,9) If not found, return response code 6316.
Response Code	6316
Response Description	Reconciliation status of "STL Complete with Manual Intervention" is out of sequence with the reconciliation status recorded in the ITL.

1026
1027

12. Reconciliation Results

Check Name	Account Type/Unit Type Totals
Check Description	The totals for account types and unit types must be consistent.
Process	Return response code 6410.
Response Code	6410
Response Description	There is an inconsistency in the totals by account type and unit type.

Check Name	Account Type/Unit Type Unit Blocks
Check Description	The registry and ITL unit blocks for a specified account type and unit type must be consistent.
Process	Return response code 6420.
Response Code	6420
Response Description	The unit blocks for the specified account type and unit type are inconsistent.

Check Name	Snapshot DateTime Validity
Check Description	The DateTime for reconciliation action proposed by STL must be in the future.
Process	Return response code 6440.
Response Code	6440
Response Description	The date and time for the proposed reconciliation action occurs in the past. Snapshot DateTimes must be in the future.

Check Name	Ongoing Reconciliation
Check Description	A reconciliation action cannot be initiated at the registry because there is already an ongoing action.
Process	Return response code 6450.
Response Code	6450
Response Description	A reconciliation action cannot be initiated at the registry because there is already an ongoing action.