DATA EXCHANGE STANDARDS FOR
REGISTRY SYSTEMS UNDER THE KYOTO PROTOCOL

DRAFT
TECHNICAL DESIGN SPECIFICATION

ANNEXES

Non-paper

8 June, 2004

[This page intentionally left blank.]

Table of Contents

Annex A
Glossary of Terms

Figure A1: Glossary of Terms

Term	Definition
AAU	Assigned amount units. These are tradable units derived from an Annex I Party's emissions target under the Kyoto Protocol. They may be counted by Annex I Parties towards compliance with their emissions target and are equal to one tonne of carbon dioxide equivalent gases.
Account	An account is used to partition a registry and can hold units. There are three accounts types: holding account, cancellation account and retirement account.
Accuracy	Condition in which information is not modified randomly by the software system.
Acknowledgement	An acknowledgement is the communication that is returned by a Web service (located at either the Transaction Log or at a registry) that a message has been successfully received and the transmission was successful. The acknowledgement occurs before the message is evaluated in any way other than format checks and minimum version requirements.
Administrator	A role to configure and maintain a software system. Configuration can range from system set-up to amending data and parameters within the system.
Annex I Party	A Party to the UNFCCC listed in Annex I to the UNFCCC. These are industrialized countries, including those with economies in transition.
Article	An Article of the Kyoto Protocol.
Attribute	Identifier for a piece of information.
Audit	Checking of recorded data.
Authentication	The process to confirm the identity of a user.
Authorization	The process to verify a permission to do something.
Cancellation	Cancellation is the action taken by the Transaction Log to cancel a proposed transaction when no response has been received from a registry within 24 hours.
CER	Certified Emission Reduction unit. These are tradable units generated by Projects in non-Annex I Parties under the CDM. They may be counted by Annex I Parties towards compliance with their emissions target and are equal to one tonne of carbon dioxide equivalent gases.
CDM	Clean Development Mechanism under Article 12 of the Kyoto Protocol. Projects in developing countries under the CDM result in reduced emissions, or enhanced removals, in a host non-Annex I Party and generate CERs.
CDM Executive Board	The board supervising the CDM. It is serviced by the secretariat.
CDM Project	A project identifier under the Clean Development Mechanism.

(cont.)

Figure A1: Glossary of Terms (cont.)

Term	Definition
CDM Registry	The registry established by the CDM Executive Board on behalf of Non-Annex I Parties hosting CDM Projects. It is to ensure the accurate accounting of transactions of CERs, lCERs, and tCERs by those Parties.
Certificate Authority	Provides a digital certificate for site-to-site Authentication. Is provided commercially by several external vendors. These vendors verify identity and issue certificates which can be used to positively identify an organization and encrypt data communications between the organization and other certificate holders.
Commitment Period	A specified period in which an Annex I Party is to show compliance with its emissions target. The first Commitment Period for the Kyoto Protocol is from 2008 to 2012.
Communications Hub	The central communication component integrated in the ITL, through which all registries and the ITL communicate.
Component	A component is a group of programming functions that perform related tasks.
Conference of the Parties (COP)	The supreme decision-making body under the UNFCCC. Attended by delegations from all state Parties to the UNFCCC. COP9 will be held in Milan, Italy, from 1-12 December 2003.
Customisation	Configuration of systems toward specific user needs within certain boundaries.
Denial of Service Attack	A very high number of requests in very short period aimed at a software system with the goal of achieving an overload and crash of that software system.
Digital Certificate	Provided by the Certificate Authority to ensure authentication of documents
Downtime	The time in which a software system is not available for use.
Emissions trading	The trading of units which may count towards compliance by Annex I Parties with their emissions targets. Emissions trading is provided for under Article 17 of the Kyoto Protocol. Domestic (e.g. UK) and regional (e.g. EU) emissions trading schemes are also being established.
Encryption	A way of protecting data from unauthorized access.
Entities	Legal entities authorized by a government to participate in emissions trading or joint implementation Projects. Private and/or public entities involved in the CDM. Such entities may be from public, private or non-governmental sectors.
ERU	Emission reduction units. These are tradable units generated by Projects in Annex I Parties under joint implementation. They may be counted by Annex I Parties towards compliance with their emissions target and are equal to one tonne of carbon dioxide equivalent gases.
Exchange mechanism	System for exchanging data.

(cont.)

7

Figure A1: Glossary of Terms (cont.)

Term	Definition
Finalisation	Finalisation is the action taken by a registry to complete a transaction which has been validated by the Transaction Log.
Function	A specific section of programming code within a component which performs a specific task.
Functional requirement	Requirement for which the quality test result is binary (e.g., yes/no or right/wrong).
GUI	Graphical User Interface.
Inconsistency	A difference found through a comparison of at least two different data sets.
Integrity	Data cannot be modified by any party not authorized to do so.
Invalidation	An invalidation is a finding by the Transaction Log that a message does not conform to the messaging requirements (including data formats, identifiers, etc.) in this Technical Specification.
Joint Implementation Project	Project identifier by an Annex I Party under the Joint Implementation Mechanism of the Kyoto Protocol.
Kyoto Protocol	Allied agreement to the UNFCCC containing emission reduction targets for Annex I Parties.
Logging	Functionality of a software system that stores information on the system for auditing and tracking.
Major Version Number	A major version number is the number assigned to the Technical Specification for Data Exchange Standards for purposes of identifying a specific set of technical requirements. The major version number changes only when a change in the Technical Specifications requires programming changes in a registry.
Minor Version Number	A minor version number is the number assigned to identify version changes in the Technical Specifications for Data Exchange which do not require programming changes within registries. These changes may involve response code table updates, for example.
Message	Electronic transmission of data generated upon a manual instruction.
National Registry	A registry established by an Annex I Party.
Non-Annex I Party	A Party to the UNFCCC which is not listed in Annex I to the UNFCCC. These are developing countries.
Non-functional requirement	Requirement for which the quality test result is measure or a score (e.g., from 1-10 or high/medium/low)
Notification	Electronic transmission of data generated automatically upon receipt of a transmission (e.g., a message or another notification).

(cont.)

8

Figure A1: Glossary of Terms (cont.)

Term	Definition
OLA	Operation Level Agreements
Party	A state that has ratified the Kyoto Protocol.
Process	The business area or category of interaction between registries and the Transaction Log. The primary processes are unit issuance, unit conversion, external transfers, internal transfers (including trades, cancellations and retirements), unit carry-overs and reconciliation. In addition, there is a Transaction Log Administration process which addresses the need to manage overall timing and communications failures.
Protocol	Formal rules describing how to transmit data.
Question of Implementation	A problem identified in the review of a Party's emissions inventory or other information submitted by a Party in the context of the Kyoto Protocol.
Reconciliation	The process by which data from different registry systems are compared and inconsistencies are resolved.
Recovery	The complete re-installation and re-configuration of a software system from scratch.
Registry	A software system for the accounting of transactions in AAUs, RMUs, ERUs and CERs. Includes national registries and the CDM registry.
Registry system	Generic term for national registries, the CDM registry and the Transaction Log.
Release Process	Process that describes how a new piece of software is deployed to the system in operation.
Removal	Removals of greenhouse gases from the atmosphere through LULUCF activities. Such removals may lead to the generation of RMUs, CERs or ERUs. They are the "opposite" of emissions of greenhouse gases.
Response	A response is the information sent following the processing of a proposed transaction. Typically the response includes the transaction ID, an indicator that the proposed transaction was successful or unsuccessful, and, if unsuccessful, the response code(s) providing the reason for the failure.
RMU	Removal units. These are tradable units generated on the basis of removals of greenhouse gases from the atmosphere through LULUCF activities under Articles 3.3 and 3.4 of the Kyoto Protocol. They may be counted by Annex I Parties towards compliance with their emissions target and are equal to one tonne of carbon dioxide equivalent gases.
Robust	A characteristic of a software system that describes the extent to which it is protected from loss of service or data integrity.
Role	A role is a set of permissions for functions that a person is allowed to perform. A role may be assigned to a user (person) or a group.

(cont.)

9

Figure A1: Glossary of Terms (cont.)

Term	Definition
Scalability	The ability of a software system to handle higher workload then initially planned without modifying the program code.
Secretariat	Secretariat to the UNFCCC
SLA	Service Level Agreement.
SOAP	Simple Object Access Protocol
Stage	A uniquely identifiable step in a data exchange sequence.
Status	A characteristic given to transaction or stage of reconciliation (e.g., initiated, terminated, completed)
Supplementary Program	An emissions tracking program for GHG emissions which operates as a complementary program to the Kyoto Protocol and shares communications with Kyoto Protocol participants through the ITL Communications Hub.
TCP	Transmission Control Protocol.
Termination	Termination is the action taken by a registry to end a proposed transaction which has been determined to be invalid, for which a discrepancy has been identified, or for which the allowable response time has lapsed.
Transaction	An operation applied to AAUs, RMUs, ERUs and CERs (e.g., issuance, transfer, cancellation, retirement, carry-over, replacement, or expiry date change).
Transaction log	An electronic database established by the Secretariat to monitor the validity of transactions between registries.
Transaction status	A transaction must have one of the following statuses: proposed, checked (no discrepancy), checked (discrepancy), final, completed, cancelled, terminated, or rejected.
True-up period	The period from the end of the Commitment Period (2012) until 100 days after the completion of the Kyoto Protocol reviews of emissions information relating to the Commitment Period. Transfers of units may continue to take place during this period. The true-up period may therefore last until some time in 2015.
UML	Unified Modeling Language. Notation standard for describing software systems.
Unavailable status	Units which are involved in transactions that have been proposed, received by the Transaction Log, and are waiting for a response from either another registry (for an external transfer) or the proposing registry are “unavailable” for other transfer. These units are placed in a queue. Similarly, a unit involved in either a discrepancy or inconsistency is placed in the queue and is unavailable until the inconsistency is resolved or the transaction involving the discrepancy is “complete.”

(cont.)

Figure A1: Glossary of Terms (cont.)

Term	Definition
UNFCCC	United Nations Framework Convention on Climate Change. This is the framework treaty to which the Kyoto Protocol is allied.
Unit	Generic term for AAUs, RMUs, ERUs, lCERs, tCERS and CERs.
Universal time	Equivalent to Greenwich Mean Time (24 hour clock).
User	A person (human being) who interacts with a system.
User Acceptance Test	A test performed by a user of the system against a set of predefined test cases.
User Interface	The interface used by a person to interact with an application.
Virus	A software program that harms software systems or other software programs.
VPN	Virtual Private Network.
Web Service	Programming functionality to secure encrypted messages or information over the Internet, protecting them from modification or interception in transit.
WSDL	Web Service Description Language.
XML	Extensible Markup Language. Standard used for structured data storage.

12

Annex B
Web Service Operations and Functions for Transaction Processing

Annex B
Web Service Operations and Functions for Transaction Processing

1. Introduction

This annex contains a specification for the Web services and functions the registries must implement in order to establish communication with the ITL for transaction processing. Each Web service operation and suggested function is detailed in order to specify how the registry passes information to and from the ITL. Accordingly, the data elements that are passed into and out of a Function accessible through Web services are defined.

Some of these operations and functions are exposed to the public through Web services and others are functions that operate internally to the registry. In general, the input and output parameters have not been specified for functions that operate internally to a system. The design of the Web services public functions, including how they store and process data, may vary from registry to registry.

Only Web services and operations that are required for a registry are listed in this annex. For a complete list of the corresponding and interacting Web service operations and functions performed by the ITL, reference the ITL Technical Specification document, Annex E.

2. Objects and Structures

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

2.1 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.

2.2 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 (optional)
- TransferringRegistryIdentifier As String
- TransferringRegistryAccountType As Integer
- TransferringRegistryAccountIdentifier As Long (optional)
- AcquiringRegistryIdentifier As String (optional)
- AcquiringRegistryAccountType As Integer
- AcquiringRegistryAccountIdentifier As Long (optional)
- NotificationIdentifier As Integer (optional)
- TransactionBlocks (array) As TransactionUnitBlockObject

This object contains a SuppTransactionType element which is used by the Supplementary Transaction Log for the European Commission GHG Trading Program.

2.3 TransactionUnitBlock Object

The functions below reference a TransactionUnitBlock Object to obtain the unit blocks associated with a transaction. Often, an array of TransactionUnitBlock Objects is called for because more than one block can be associated with a transaction. The structure of the TransactionUnitBlock Object is as follows:

Structure TransactionUnitBlock
- UnitSerialBlockStart As Long
- UnitSerialBlockEnd As Long
- OriginatingRegistryIdentifier As String
- UnitType As Integer
- SuppUnitType As Integer (optional)
- OriginalCommitPeriod As Integer
- ApplicableCommitPeriod As Integer
- LULUCFActivity As Integer (optional)
- ProjectIdentifier As Integer (optional)
- Track As Integer (optional)
- BlockRole As String (optional)
- TransferringRegistryAccountType As Integer (optional)
- TransferringRegistryAccountIdentifier As Long (optional)
- TransferringRegistryIdentifier As String (optional)
- AcquiringRegistryAccountType As Integer (optional)
- AcquiringRegistryAccountIdentifier As Long (optional)
- AcquiringRegistryIdentifier As String (optional)
- YearInCommitmentPeriod As Integer (optional)
- InstallationIdentifier As Long (optional)
- ExpiryDate As DateTime (optional)

This object contains a SuppUnitType, Year in Commitment Period, and Installation Identifiers which are used by the Supplementary Transaction Log for the European Commission's GHG trading program.

2.4 UnitBlock_Identity Object

The UnitBlock_Identity Object contains the attributes that uniquely identify a unit block and is generally returned as part of a notification to a Registry in conjunction with a CheckResponse Object.

Structure UnitBlock_Identity Object
- UnitSerialBlockStart As Long
- UnitSerialBlockEnd As Long
- Originating Registry Identifier As String

2.5 Check Response 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
- ResultIdentifier As Integer
- ResponseCode As Integer

2.6 Evaluation Result Object

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

Structure EvaluationResultObject
    CheckResponseObject
    UnitBlocks (array) As UnitBlock_Identity Object

3. Specified Functions

The following functions and Web service operations will be implemented to carry out the duties of a registry. They are listed in alphabetical order.

Figure B1: AcceptNotification (Web service)

Purpose
This is the HTTP SOAP request that registries and the ITL use to send notifications regarding transactions in process. This Web service is hosted on both the ITL and the registry. See Annex K for specifications on the WSDL requirements.
Inputs
TransactionObject, EvaluationResultObject
Process
This Web service performs preliminary checks on the content of the XML message before processing.
Outputs
Result_Identifier
Recommended Functions
Data_Integrity_Check

159 **Figure B2: AcceptProposal (Web service)**
160

Purpose
This is the HTTP SOAP request that the ITL uses to receive proposals from registries. Registries also must implement this service to receive proposals for external transfers from the ITL.
Inputs
TransactionObject
Process
This Web service should perform preliminary checks on the content of the XML message before logging the message and writing contents of the message to a file.
Outputs
Result_Identifier
Recommended Functions
Data_Integrity_Check Write_To_File Write_To_Message_Log

161

Figure B3: Check_Version (Function)

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 DES. Major version numbers must match or any forthcoming transactions to the ITL will be rejected. Messages with an outdated minor version are still processed, although a warning is issued. This is an optional function for registries.
Inputs
MajorVersion, MinorVersion
Process
Compare MajorVersion. If not a match, take corrective actions to update Registry system. This requires compliance with change management processes as defined on the ITL intranet website. Compare MinorVersion. If not a match, then be prepared to update system with minor patches or enhancements as recommended on the ITL intranet website.
Outputs
CheckResponseObject

Figure B4: 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. This is an optional function for registries.
Inputs
TransactionObject
Process
For the following processes, ensure that the incoming message meets the following criteria. See Annex E for a list of appropriate response codes to return for data integrity errors.
Outputs
CheckResponseObject TransactionObject

Figure B5: Finalise_Transaction (Function)

Purpose
When a transaction is complete, this function is called to update the unit holding records of the registry.
Inputs
TransactionObject
Process
This function evaluates the transaction type to determine the data which must be updated. If the transaction type = 1 (Issuance) or 10 and the supplementary transaction type = 51 or 54, insert records representing unit blocks and accounts, as appropriate. If the transaction type = 2, 7 or 8 (Conversion, Carry-over, or Expiry Date Change), the units are free to be used in a trade. Registries will commit the change of unit type and project ID (for Conversion), the applicable commitment period (for Carry-over), or the expiry date (for Expiry Date Change). If the transaction type = 3, 4, 5 (External, Cancellation, Retirement) or (10 and the supplementary transaction types = 53, 21, 55, 01, 02, 41), the units are free to be used in a trade. Registries will commit the transfer ownership of the units to the acquiring party and accounts. If the transaction type = 6 (Replacement), the units are free to be used in a trade. Registries will commit the transfer to the replacement account and record the relationship between the two types of blocks.

Figure B6: Generate_Proposal (Function)

Purpose
This function creates the TransactionObject and corresponding TransactionUnitBlock for submission of a proposed transaction to the ITL. This function invokes AcceptProposal() on the ITL.
Inputs
Outputs
TransactionObject, TransactionUnitBlock
Call(s)
Write_Transaction Write_Transaction_Block Write_Transaction_Status

Figure B7: Preliminary_Checks (Function)

Purpose
The registry 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 is ready to be processed.
Inputs
TransactionObject, ReconciliationObject
Process
If this message contains a transaction, call Write_To_File to record the contents of the message.
Outputs
Result_Identifier
Call(s)
Check_Version Write_To_File

Figure B8: Update_Units (Function)

Purpose
This function will record the transaction as final and update other records relating to the transaction, as appropriate.
Inputs
TransactionObject, TransactionUnitBlockObject
Process
If the transaction type is External (3), the transaction is recorded as final and the units are removed from the appropriate accounts. If transaction type is Carry-over (4), then the Applicable Commitment Period for the unit is updated to the current commitment period. If the transaction type is Conversion (2), and the unit was not previously (2) RMU, then the Unit Type is updated to (3) for ERU and the Project ID is updated. If the transaction type is Conversion (2), and the previous unit type was (2) RMU, the Unit_Type is updated to (4) for ERU converted from RMU and the Project ID is updated. If the transaction type is Cancellation (4) or Retirement (5), then transaction is recorded as final and the units moved to the appropriate account(s). If the transaction type is Expiry Date Change (8), update the expiry 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

183 **Figure B9: Validate_Proposal (Function)**
184

Purpose
This function is used only by acquiring registries to validate external transfers. This function could call other functions on the registry to perform additional checks.
Inputs
TransactionObject, TransactionUnitBlockObject
Process
Incoming proposals are evaluated and processed. Data integrity checks should be applied to proposed transaction. Log and record the transaction, applicable blocks and the transaction status. If the proposal is accepted or rejected, log the new transaction status and call the ITL AcceptNotification Web service, submitting the TransactionObject and TransactionUnitBlockObject with the transaction status and any applicable response codes.
Calls
Data_Integrity_Check Write_To_Message_Log Write_Transaction Write_Transaction_Block Write_Transaction_Status AcceptNotification

185

Figure B10: Write_To_File (Function)

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.
Outputs
Result_Identifier
Call(s)

Figure B11: Write_To_Message_Log (Function)

Purpose
This function will append a record to the Message Log. The Message Log table records the history of all message exchange. The contents of any HTTP SOAP Request received involving a transaction 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)

192 **Figure B12: Write_Transaction (Function)**
193

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

194

Figure B13: Write_Transaction_Block (Function)

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

198 **Figure B14: Write_Transaction_Status (Function)**
199

Purpose
This function will insert a record into the Transaction Log History table.
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 from TransactionObject.
Outputs
Transaction Status ID, unique identifier for new Transaction Status History record, Result_Identifier
Call(s)

200
201
202

Annex C
Web Services and Functions for Reconciliation

Annex C
Web Services and Functions for Reconciliation

1. Introduction

This annex contains a specification for the Web services and functions the registries must implement in order to establish communication with the ITL for reconciliation. Each Web service operation and suggested function is detailed in order to specify how the registry passes information to and from the ITL. Accordingly, the data elements that are passed into and out of a function accessible through Web services are defined.

Some of these operations and functions are exposed to the public through Web services and others are functions that operate internally to the registry. In general, the input and output parameters have not been specified for functions that operate internally to a system. The design of the Web services public functions, including how they store and process data, may vary from registry to registry.

Only Web services and operations that are required for a registry are listed in this annex. For a complete list of the corresponding and interacting Web service operations and functions performed by the ITL, reference the ITL Technical Specification document, Annex E.

2. Objects and Structures

The following sections explain the conventions and structured data storage utilised in the specifications for each function. Also consult the TransactionObject, TransactionUnitBlockObject, UnitBlockIdentityObject, and CheckResponseObject defined in Annex B.

2.1 Reconciliation Object

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

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

2.2 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
- ReconciliationSnapshotDate As DateTime
- HoldingRegistry As String
- AccountType As Integer
- AccountIdentifier As Integer
- UnitType As Integer
- SupplementaryUnitType As Integer
- UnitCount As Integer

275
276

Figure C1: Calculate_Totals (Function)

Purpose
This function will calculate the total number of units held at the registry in response to a request from the ITL. The totals will be calculated by querying the snapshot data that was stored at the requested date and time for the reconciliation. The ITL may request totals only for units held by a specific account type or unit type. If no specific requests are made, the totals should be calculated for each distinct combination of account type and unit type.
Inputs
ReconciliationObject, Holding Account Type, Unit Type
Process
If no limiting criteria were passed into the ProvideTotals service, Calculate the number of units held grouped by account type and unit type. Below is an example of an SQL query to calculate the totals. Select Account Type, Unit Type, Sum(end_block - start_block + 1) as Unit Totals From Reconciliation Snapshot Data Where Reconciliation ID = current reconciliation action ID Group By Account Type, Unit Type If limiting criteria were passed into the ProvideTotals service, Calculate the number of units held by the specified account type or unit type. Below is an example of an SQL query to calculate these totals. Select Account Type, Unit Type, Sum(end_block - start_block + 1) as Unit Totals From Reconciliation Snapshot Data Where Reconciliation ID = current reconciliation action ID and Account Type = account type specified by ITL and Unit Type = unit type specified by ITL Group By Account Type, Unit Type Store the totals returned in an array of the TotalObject, and call the ReceiveTotals Web service method on the ITL.
Outputs
TotalsObject array

277 **Figure C2: Close_Reconciliation_Action (Function)**
278

Purpose
This function will update the registry's Reconciliation Log with the end date of a reconciliation action.
Inputs
ReconciliationObject
Process
Update Reconciliation Log tables so that Reconciliation End Date = date and time when ITL confirms reconciliation compliance. This information is received from the ReceiveReconciliationResult Web service operation.
Outputs
ResultIdentifier

279

Figure C3: Generate_Audit_Trail (Function)

Purpose
This function identifies past transactions involving unit blocks that are in question. In order to identify the source of an inconsistency, the ITL will request data for all the transactions that occurred within a specified time period for the unit blocks in question. This function will accumulate a list of transactions by querying the transaction log for the specified time period and send the audit trail to the ITL.
Inputs
ReconciliationObject, UnitBlock_IdentityObject, Begin Transaction Time, End Transaction Time
Process
If no limiting criteria were passed to the ProvideUnitBlocks service, Generate a complete list of unit blocks. Below is an outline of an SQL query to generate the list of unit blocks. Select Account Type, Unit Type, Originating Registry Code, Start Block, End Block From Reconciliation Snapshot Data Where Reconciliation ID = current reconciliation action ID If limiting criteria were passed to the ProvideUnitBlocks service, Generate a list of unit blocks that meet the specified criteria. Below is an outline of an SQL query to generate the list of unit blocks. Select Account Type, Unit Type, Originating Registry Code, Start Block, End Block From Reconciliation Snapshot Data Where Reconciliation ID = current reconciliation action ID and Account Type = account type specified by ITL and Unit Type = unit type specified by ITL Store the Unit Blocks identified in an array of the UnitBlockObject, and call the ReceiveUnitBlocks web service method on the ITL.
Outputs
UnitBlockObject array
Call(s)

Figure C4: Generate_Unit_Blocks (Function)

Purpose
This function identifies unit blocks requested by the ITL for comparison during reconciliation. This function will accumulate a list of unit blocks by querying the snapshot unit block data that was stored at the requested date and time for the reconciliation. The ITL may request only the unit blocks of a specific Unit Type and/or held by a specific Account Type be sent. If no criteria are specified, all unit blocks should be sent to the ITL.
Inputs
ReconciliationObject, Holding Account Type, Unit Type
Process
Query the Transaction Log and the Transaction Block tables and return all available information for the specified units and time period. Below is an outline of an SQL query to generate the audit trail. Select Transaction Number, Transaction Date, Transaction Type, Transferring Registry, Acquiring Registry, Transferring Account, Acquiring Account Originating Registry, Start Block, End Block, Unit Type, Applicable Commit Period, Original Commit Period, Project ID, LULUCF Activity Code, Track ID, Expiry Date, From Transaction Log, Transaction Block Where Transaction Date >= requested begin transaction time and Transaction Date <= requested end transaction time and Originating Registry = requested originating registry and ((Start Block >= requested start block and Start Block <= requested end block) or (End Block >= requested start block and End Block <= requested end block) Store the audit trail to a TransactionObject array, and call the ReceiveAuditTrail web service method on the ITL.
Outputs
TransactionObject array
Call(s)

Figure C5: ProvideAuditTrail (Web service)

Purpose
This is the HTTP SOAP request that the ITL uses to request audit trail data from a registry.
Inputs
ReconciliationObject
Process
This function receives a request from the ITL to provide a transaction history for a specified set of units within a specified timeframe. See Section 5 for detailed information on the reconciliation process.

Figure C6: ProvideTotals (Web service)

Purpose
This is the HTTP SOAP request that the ITL uses to initiate a request for total number of units held by each registry.
Inputs
ReconciliationObject
Process
This service receives a request for the total number of units held by each registry. The request may be for a specified account type, unit type, or a combination of the two. If no limiting criteria are passed, the request is for all totals, grouped by account type and unit type. See Section 5 for detailed information on the reconciliation process.

293 **Figure C7: ProvideUnitBlocks (Web service)**
294

Purpose
This is the HTTP SOAP request that the ITL uses to initiate a request for the unit blocks held at a registry.
Inputs
ReconciliationObject
Process
This service receives a request for the unit blocks held at a registry. The request may be for a specified account type, unit type, or a combination of the two. If no limiting criteria are passed, the request is for all unit blocks. See Section 5 for detailed information on the reconciliation process.
Outputs
CheckResponseObject
Call(s)
Preliminary_Checks

295
296
297
298

299 **Figure C8: ReceiveReconciliationResult (Web service)**
300

Purpose
This is the HTTP SOAP request that the ITL uses to inform the registry that it has finished processing a reconciliation action.
Inputs
ReconciliationObject
Process
This Web service receives the result of the ITL's reconciliation analysis. Registries will record the results of the analysis in their Reconciliation Logs. The ITL will only call this service when the reconciliation action is over.
Outputs
CheckResponseObject
Call(s)
Close_Reconciliation_Action Write_To_Reconciliation_Log Write_To_Reconciliation_Status

301

302 **Figure C9: Start_Reconciliation (Function)**
303

Purpose
This function opens a reconciliation action based on an ITL request for the registry to send data to the ITL for a specified time. The registry must take a "snapshot" of the data at that time and provide data to the ITL from the snapshot.
Inputs
ReconciliationObject
Process
Insert a new record into the reconciliation log indicating the beginning of a new reconciliation action. At the specified time, get a "snapshot" of the unit block data. The fields needed in the snapshot for each unit block include: • Holding Account Type Code • Unit Type Code • Initiating Registry • Start Block Number • End Block Number The Reconciliation ID should also be stored. This data should be preserved at least until the reconciliation action has closed.
Outputs
Call(s)

304

305 **Figure C10: Write_To_Reconciliation_Log (Function)**
306

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 • Recon_Log_Comment = input comment • Recon_Action EndDateTime = input reconciliation end DateTime • Recon_Snapshot_DateTime = input reconciliation snapshot DateTime • Recon_Phase_Code = input reconciliation phase
Outputs
ResponseObject
Call(s)

307

Figure C11: Write_To_Reconciliation_Status (Function)

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_Comment = input comment • Recon_Log_DateTime = system time
Outputs
ResponseObject ReconciliationObject
Call(s)

Annex D
Web Services for Administrative Processes

Annex D
Web Services for Administrative Processes

1. Introduction

This annex contains a specification for the Web services and functions the registries must implement in order to establish communication with the ITL for administrative processes. Each Web service operation is detailed in order to specify how the registry passes information to and from the ITL. The design of the Web services public functions, including how they store and process data, may vary from registry to registry.

Only Web services and operations that are required for a registry are listed in this annex. For a complete list of the corresponding and interacting Web service operations and functions performed by the ITL, reference the ITL Technical Specification document, Annex E.

2. Objects and Structures

The Web services defined for administrative processes make extensive use of the TransactionObject and the TransactionUnitBlockObject. See Annex B for a definition of these objects.

Figure D1: AcceptMessage (Web service)

Purpose
This HTTP SOAP service receives incoming messages forwarded from the ITL. This Web service may be used to receive messages from the ITL or another registry. The Content input element is flexible and can contain a large amount of text. This allows the function to act as a generic service for registries to send messages to each other. The ITL will call this Web service to inform the registry about issues identified by administrative and cleanup processes on the ITL. This Web service will be used by the following processes. 1. Inform the registry of Outstanding Units at the end of the Commitment period. A process on the ITL that runs after a commitment period ends will identify all units that have not been cancelled or carried-over. Each registry that holds one of these units will receive a message through this Web service that action must be taken on these units within 30 days. The outstanding units will be specified in the TransactionUnitBlockObject array. 2. Inform the registry of units that are about to expire. The ITL will identify all lCERs and tCERs that will expire within 30 days. Each registry holding one of these units will be notified through this Web service. The affected units will be listed in the TransactionUnitBlockObject array. The registry should either cancel or replace these units within 30 days. 3. Inform the registry of lCERs that may not be traded due to a lack of certification report for the project associated with the lCERs. The affected units will be listed in the TransactionUnitBlockObject array. Any transaction containing one of these units will be rejected by the ITL, except for a cancellation transaction. 4. Inform the registry the need to replace or cancel units due to a Reversal in Storage at a Project. The ITL will send a message that all lCERs held by the registry cannot be traded until the registry replaces a specified number of units. When the registry proposes a replacement transaction or a cancellation transaction in response, the transaction proposal should include the Notification ID included in the original message from the ITL. Through this ID, the ITL will track the registry's progress toward replacing the specified number of units. Once the specified number has been achieved, the ITL will again use this Web service to inform the registry that it has complied and that the remaining lCERs are free to be transferred.
Inputs
From, To, MajorVersion, MinorVersion, MessageContent, MessageDateTime, NotificationID, TransactionUnitBlockObject array
Process
Upon receipt of an incoming text message from the AcceptMessage Web service, the system administrator shall be alerted of its arrival.
Outputs

Figure D1: AcceptMessage (Web service) (cont.)

Call(s)

348 **Figure D2: GetTransactionStatus (Web service)**
349

Purpose
This is the HTTP SOAP request registries use to retrieve the current status of a transaction at the ITL.
Inputs
Transaction ID
Process
This function will query the ITL database and return the latest status of a specified transaction. It will also return the date and time the status was last updated.
Outputs
TransactionStatus TransactionStatusUpdateDateTime
Call(s)

350

Figure D3: ProvideTime (Web service)

Purpose
This is the HTTP SOAP request that the ITL uses to request audit trail data from a registry.
Inputs
From, To, MajorVersion, MinorVersion
Process
The ITL will periodically request a registry to return the current time in order to ensure that the system time is in synchronization with the ITL.
Outputs
SystemTime ResultIdentifier ResponseCodes
Call(s)

Annex E
List of Checks and Response Codes for Transaction Processing

1. Summary of Response Specifications

The ITL sends responses to requests or proposals indicating the success or failure of a transaction. Responses can be "simple" with only two pieces of data to report, or "complex" containing large record sets of data. Simple responses, which check for a minimal version control test, merely send back an indication that the message was received successfully and that the message is queued for processing. Complex responses are messages that are sent after a transaction has been evaluated by the ITL. If the transaction or the validation process was unsuccessful, the Transaction Log sends a message containing response codes providing the reason for the failure.

Each registry shall have a response lookup table and the capability to receive and import updates to this table to support change management specifications described in Section 8. Registries shall use programming techniques which facilitate updates to codes in these tables without programming changes whenever possible.

A registry may not change the number associated with a response or the meaning of the response description. Each registry may elect to translate the response description into a different language for display or reporting purposes only. A registry may request that a response be added to the system by submitting a written request to the ITL administrator (or through an alternative process).

Response codes are assigned to the following range of numbers by category.

Figure E1: Check Categories

Category	Response Code Range	Category Description	Action Upon Failure
Version and Authentication	1000 - 1299	Checks to authenticate sender and to validate version of DES during preliminary processing.	Message returned with response codes or HTTP Soap Error. Message not placed into message queue (unless only a minor version inconsistency is identified).
Message Viability	1300 - 1399	Checks to determine whether the message is viable when processed from the queue.	Message returned with response codes. Message not logged in the Transaction_Log table.
Registry Validation	1500 - 1599	Checks to validate status of registry during queue processing.	Message returned with response codes. Message not logged in the Transaction_Log table.
Data Integrity	2000 - 2999	Basic checks of data content including numeric ranges and validity of codes during queue processing.	Message returned with response codes. Message not logged in the Transaction_Log table.
Message Sequence for Registry Messages	3000 - 3499	Checks to validate message order and transaction status.	Message returned with response codes. Message not logged in the Transaction_Log table.
Message Sequence for STL Messages	3500 - 3999	Checks to validate message order and transaction status.	Message returned with response codes. Message not logged in the Transaction_Log table.

(cont.)

Figure E1: Check Categories (cont.)

Category	Response Code Range	Category Description	Action Upon Failure
General Transaction Checks	4000 - 4999	Checks applicable to all transactions involving unit blocks. These checks are applicable to all transactions	Message returned with response codes and transaction status. Message logged in the Transaction_Log table.
Transaction-specific Checks	5000 - 5899	Kyoto Protocol transaction checks specific to designated transaction types.	Message returned with response codes and transaction status. Message logged in the Transaction_Log table.
Registry Messages	5900 - 5999	Response codes generated by registries.	Response codes sent with transactions to other parties.

2. Transaction Processing Response IDs

The following responses are used in messages to support communications for the processes defined in Section 4.

Figure E2: Responses

Response Code	Check Name	Check Category	Check Description	Transaction Type
SOAP error	Certificate Check	Version and Authentication	If certificate is not recognized, message is rejected.	All
SOAP error	SOAP Identifier	Version and Authentication	Initiating Registry must be consistent with sender of SOAP message.	All
SOAP error	WSDL Check	Version and Authentication	Message must conform to WSDL.	All
1031	Major Version	Version and Authentication	If Major Version number in transaction message does not match Major Version number for DES.	All
1032	Minor Version	Version and Authentication	If Minor Version number in transaction message does not match Minor Version number for DES.	All
1301	Message Age	Message Viability	Message must be processed within 24 hours of submission.	All
1501	Identify Registry	Registry	Must be contained in Registry table.	All
1503	Initiating Registry Available for Transactions	Registry	Initiating Registry status code must be equal to zero.	All
1504	Acquiring Registry Available for Transactions	Registry	Acquiring Registry status code must be equal to zero.	All

(cont.)

Figure E2: Responses (cont.)

Response Code	Check Name	Check Category	Check Description	Transaction Type
2001	Transaction Mask	Data Integrity	Transaction ID must be comprised of a valid registry code followed by numeric values.	All
2003	Transaction Status Code	Data Integrity	Transaction status code must be valid.	All
2004	Transaction Type Code	Data Integrity	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.	All
2005	ERU, CER, tCER, lCER Check	Data Integrity	If the unit is a CER, an lCER or tCER, an ERU, or an ERU converted from an RMU, a Project Identifier must be present.	All
2006	ERU Track Check	Data Integrity	If a unit is an ERU, a valid Track must be present.	All
2007	Supplementary Transaction Type Checks	Data Integrity	The Supplementary Transaction Type Code must be blank or valid.	All
2080	Initiating Account Identifier	Data Integrity	Initiating Account Identifier must be greater than zero.	All
2082	Acquiring Account Identifier	Data Integrity	Acquiring Account Identifier must be greater than zero.	All
2083	Account Type Code	Data Integrity	Account Type must be valid.	All
2084	Unit Serial Range	Data Integrity	The Unit Serial end block must be greater than or equal to the Unit Serial begin block.	All
2085	Unit Type Code	Data Integrity	Unit Type must be valid.	All
2086	Unit Serial	Data Integrity	Unit Serial start block and Unit Serial end block must have a value greater than zero.	All
2087	Supplementary Unit Type Code	Data Integrity	Supplementary Unit Type Code must be valid.	All
2089	Notification ID/Registry	Data Integrity	The Notification ID must be one that was sent to the registry.	All
2090	LULUCF Code	Data Integrity	LULUCF activity code must be present in LULUCF Activity Code table.	All
2091	Commitment Period	Data Integrity	Original or applicable Commitment Period must be less than 10.	All

(cont.)

403 **Figure E2: Responses (cont.)**
404

Response Code	Check Name	Check Category	Check Description	Transaction Type
2096	Track Code	Data Integrity	Track must be blank, 1 or 2.	All
2098	Transaction Status DateTime	Data Integrity	Transaction Status DateTime must be between 1/1/05 and current date.	All
2100	RMU Check	Data Integrity	If a unit is an RMU, a valid LULUCF code must also be present for the unit block.	All
3001	Transaction Identifier Does Not Exist	Sequence	Transaction ID does not exist and transaction status = "Terminated," "Completed," or "Accepted."	All
3002	Transaction Status Out of Sequence for Prior Completed Status	Sequence	Transaction status = "Completed" and prior transaction status = "Completed."	All
3003	Non-external Accepted Status	Sequence	Check for transaction status = "Accepted" for Non-external transaction.	All
3004	Transaction Status Not Valid	Sequence	Transaction status = "Checked (No Discrepancy)," "Checked (Discrepancy)," "STL Checked (No Discrepancy)," or "STL Checked (Discrepancy)."	All
3005	Transaction Status Out of Sequence for Prior STL Discrepancy Status	Sequence	Transaction status = "Completed" and prior transaction status = "STL Checked (Discrepancy)."	All
3006	Transaction Status Out of Sequence for Prior Cancelled Status	Sequence	Transaction status = "Completed" and prior transaction status = "Cancelled."	All
3007	Transaction Status Out of Sequence for Prior Terminated Status	Sequence	Transaction status = "Completed" and prior transaction status = "Terminated."	All
3008	Transaction Status Out of Sequence for Prior Checked Discrepancy Status	Sequence	Transaction status = "Completed" and prior transaction status = "Checked (Discrepancy)."	All
3009	Transaction ID Not Unique	Sequence	Transaction status = "Proposed" and Transaction ID already exists.	All
4001	Units are Unavailable	General Transaction	All units identified in the transaction must be available for transaction (i.e., not involved in another ongoing transaction.)	All
4002	Units are Cancelled	General Transaction	Cancelled units cannot be the subject of further transactions.	All

(cont.)

405

406 **Figure E2: Responses (cont.)**
407

Response Code	Check Name	Check Category	Check Description	Transaction Type
4003	Units are Retired	General Transaction	Retired units cannot be the subject of further transactions.	All
4004	Units Have Inconsistencies	General Transaction	Units identified in the transaction cannot have existing reconciliation inconsistencies.	All
4005	Registry Holds Units	General Transaction	Units identified in transaction must be held by Initiating Registry.	All
4006	Current Commitment Period	General Transaction	Any unit in a proposed transaction must have an applicable Commitment Period identifier consistent with the current Commitment Period (including its true-up period) or for the next Commitment Period.	All
4007	Frozen Units	General Transaction	All units in the transaction must not be frozen due to a determination about a Project by the CDM Executive Board	All
5001	AAU Issuance Quantity	Transaction-specific	The number of AAUs issued must not exceed the allowable quantity for the Commitment Period (provided by the C&A database).	Issuance
5002	RMU Issuance Quantity	Transaction-specific	The number of RMUs issued for each activity type must not exceed the allowable quantity (provided by the C&A database).	Issuance
5003	RMU Issuance Timing	Transaction-specific	RMUs cannot be issued before the end of the Commitment Period, unless annual Issuance option is selected.	Issuance
5004	Consistency of Unit Type Issued for a Project	Transaction-specific	Choice of issuing tCERs or lCERs must be consistent with previous issuances for the project.	Issuance
5005	Issuing tCERs or lCERs for Project Type	Transaction-specific	tCERs and lCERs must have a LULUCF activity identifier of 1.	Issuance
5006	Expiry Date	Transaction-specific	tCERs and lCERs must have an expiry date.	Issuance
5007	tCERs Expiry Date	Transaction-specific	Expiry date for tCERs must be end of second Commitment Period.	Issuance
5008	lCERs Expiry Date	Transaction-specific	Expiry date for lCERs must be consistent with end of the crediting period of the Project.	Issuance
5009	CDM Registry Issuance of lCERs, tCERs and CERs	Transaction-specific	Only CDM Registry may issue CERs, tCERs and lCERs.	Issuance

(cont.)

408
409

Figure E2: Responses (cont.)

Response Code	Check Name	Check Category	Check Description	Transaction Type
5010	CDM Registry Issuance of Other Unit Types	Transaction-specific	CDM Registry cannot issue unit types other than CERs, lCERs, and tCERs.	Issuance
5012	CER Issuance Amount for Project	Transaction-specific	CER Issuance for each CDM Project by CDM Registry must not exceed amount specified by CDM Executive Board.	Issuance
5013	tCER Issuance Amount for Project	Transaction-specific	tCER Issuance for each CDM Project by CDM Registry must not exceed amount specified by CDM Executive Board.	Issuance
5014	lCER Issuance Amount for Project	Transaction-specific	lCER Issuance for each CDM Project by CDM Registry must not exceed amount specified by CDM Executive Board.	Issuance
5015	Issued Serial Numbers	Transaction-specific	The serial numbers issued must be unique.	Issuance
5030	Expired tCERs and lCERs Trade Restriction	Transaction-specific	If a tCER or an lCER has expired, it may not be involved in any transaction, except an Internal transfer to a Type 3 Voluntary Cancellation account.	Conversion, External, Retirement, Carry-over, Replacement
5031	Replacement Unit Trade Restriction	Transaction-specific	A unit used to replace another unit may not be involved in any other transaction.	Conversion, External, Cancellation, Retirement, Carry-over, Replacement
5032	Flagged Unit Check	Transaction-specific	If the replacement of lCERs from a CDM Project has been notified, lCERs from the Project may only be transferred to Replacement or Cancellation accounts.	Conversion, External, Retirement, Carry-over, Replacement, Expiry Date Change
5033	Unit Block Attributes	Transaction-specific	All attributes of a unit block must be consistent with ITL unit block attributes for Retirement, Cancellation, Replacement and External transactions.	Retirement, Cancellation, Replacement of External Transfers
5050	Units Available for Carry-over	Transaction-specific	Units identified in serial block must be specified as being available to be carried over.	Carry-over
5051	RMU Carry-over	Transaction-specific	RMUs may not be carried over.	Carry-over
5052	ERU Carry-over	Transaction-specific	ERUs converted from RMUs may not be carried over.	Carry-over

(cont.)

413 **Figure E2: Responses (cont.)**
414

Response Code	Check Name	Check Category	Check Description	Transaction Type
5053	lCER or tCER Carry-over	Transaction-specific	tCERs or lCERs may not be carried over.	Carry-over
5054	ERU Carry-over Limit	Transaction-specific	Total number of ERUs for Carry-over must not exceed limit.	Carry-over
5055	CER Carry-over Limit	Transaction-specific	Total quantity of CERs to be carried over must not exceed limit.	Carry-over
5056	Unit Block Attributes for Carry-over	Transaction-specific	All attributes except for Applicable Commitment Period of a unit block must be consistent with ITL unit block attributes for Carry-over transactions.	Carry-over
5101	Conversion Eligibility (Track 1)	Transaction-specific	If the unit is a Track 1 ERU, then the Party must meet all 6 eligibility criteria.	Conversion
5102	Conversion Eligibility (Track 2)	Transaction-specific	If the unit is a Track 2 ERU, then the Party must meet eligibility criteria 1, 2 and 4.	Conversion
5103	Conversion Unit Type	Transaction-specific	Units for conversion must be AAUs or RMUs.	Conversion
5104	Originating Registry for Conversion	Transaction-specific	Units for conversion must be issued by Initiating Registry.	Conversion
5105	Proposing Registry	Transaction-specific	The Initiating Registry converting AAUs or RMUs must be a national registry.	Conversion
5106	Unit Block Attributes for Conversion	Transaction-specific	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.	Conversion
5107	ERU Conversion Quantity	Transaction-specific	ERU Conversion for each Track 2 JI Project must not exceed quantity specified by the Article 6 Supervisory Committee.	Conversion
5150	CER, tCER and lCER Retirement Limit	Transaction-specific	Total quantity of tCERs and lCERs retired must not exceed limit.	Retirement
5151	CER, tCER and lCER Retirement Eligibility	Transaction-specific	If the unit is a CER, tCER, or lCER, the Party must meet all 6 eligibility criteria.	Retirement
5200	External Transfers to CDM Registry	Transaction-specific	CDM Registry can only receive transfers to Cancellation accounts.	External
5202	Acquiring Registry Eligibility for External Transfers	Transaction-specific	The Acquiring Registry must meet all 6 eligibility criteria.	External

(cont.)

415
416

Figure E2: Responses (cont.)

Response Code	Check Name	Check Category	Check Description	Transaction Type
5211	Commitment Period Reserve Level	Transaction-specific	The proposed External transfer must not result in the total registry holdings of all units to fall below the CPR level for the Transferring Registry.	External
5301	Number of Replacement Units	Transaction-specific	The number of units replaced must equal the number of replacing units.	Replacement
5302	One-To-Many Replacement Units	Transaction-specific	A transaction may not contain many-to-many relationships between replaced and replacing blocks.	Replacement
5303	Location of Replaced tCERs	Transaction-specific	Only tCERs that are held in a Retirement account or a Replacement account may be replaced.	Replacement
5304	Replacement Registry	Transaction-specific	The registry holding the units to be replaced and the replacing units must be the same.	Replacement
5305	Replacement Account Type	Transaction-specific	Only tCER Replacement accounts and lCER Replacement accounts may acquire units in a Replacement transaction.	Replacement
5306	Location of Replaced lCERs	Transaction-specific	lCERs that are held in Cancellation accounts may not be replaced.	Replacement
5308	lCER Replacement Units (Due to Expiry)	Transaction-specific	lCER Replacement accounts (unit expiry) may not acquire tCERs or lCERs.	Replacement
5309	lCER Replacement Units (Due to lack of certification or Reversal in Storage)	Transaction-specific	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.	Replacement
5310	Multiple Replacement	Transaction-specific	A unit may be replaced only once.	Replacement
5311	tCER Replacement Units (Due to Expiry)	Transaction-specific	tCER Replacement accounts (unit expiry) may not acquire lCERs.	Replacement
5401	tCER or lCER Cancellation Purposes	Transaction-specific	tCERs and lCERs may not be transferred to type 1, 2 or 4 Cancellation accounts.	Cancellation
5450	Units for Expiry Date Change	Transaction-specific	The units for Expiry Date Change must be tCERs or lCERs.	Expiry Date Change

(cont.)

Figure E2: Responses (cont.)

Response Code	Check Name	Check Category	Check Description	Transaction Type
5451	New tCER Expiry Date	Transaction-specific	The new tCER expiry date must be consistent with the end date of the second Commitment Period.	Expiry Date Change
5452	New lCER Expiry Date	Transaction-specific	The new lCER expiry date must be consistent with the end date of the renewed crediting period for the Project.	Expiry Date Change
5453	Unit Block Attributes	Transaction-specific	All attributes of a unit block except for the Expiry Date must be consistent with ITL unit block attributes for Expiry Date Change transactions.	Expiry Date Change

Figure E3: Registry Messages

Response Code	Response Description
5902	Acquiring account does not exist.
5903	Acquiring account is not eligible to receive units.
5904	Transaction inconsistent with Party policy.
5905	Transaction rejected by account holder.
5906	Account has been closed.

Annex F
Definition of Identifiers

1. Introduction

This annex provides information on the required structure of identification numbers for the core entities associated with data exchange.

2. General Rules for XML Formats

Identifiers shall be transmitted in XML format as an element tag comprised of attributes (components of the identifier). For numeric elements, leading zeros should not be included as place holders for a number shorter than the maximum length. For components which do not apply (such as project ID for an AAU), an attribute is not required.

3. Recommended Display and Report Formats

For display and reporting, it is recommended that each element of the identifier be separated by a single dash "-" and no spaces. For example, a transaction number would be represented as follows: NZ-132-1. Leading zeros would not be displayed. Please note that the separating dash is not included in the XML structure for identifiers and should not be stored data. Consistent with the requirements below, all serial numbers shall be stored as elements.

4. Serial Numbers

The serial number of a unit shall be unique throughout all registries and the ITL.

Serial numbers are defined only by registries.

Whenever possible, a set of units shall be transmitted as a unit block defined by the starting block number and the ending block number. Within a unit block, every element of the serial number must be identical except for the unique number element. Number elements within a block must be complete and consecutive.

When necessary to perform a transaction, track, record, or otherwise characterize a unit or unit block, registries or the ITL shall create multiple unit blocks from a single unit block.

Although each unit is identified uniquely by its originating Registry code and its assigned unique number and this identification should be used by registries, communication about unit(s) will conform to the Unit Block definition requirements below. For a single unit the start and end block elements contain the same value.

Figure F1: Serial Number Identifiers

Identifier	Display Order	Identifier Required for the Following Unit Types	Data Type	Length	Range or Codes
Originating Registry	1	All	Alphanumeric	3	Per country codes in ISO3166, as of 01 January 2005
Unit Type	2	All	Numeric	2	1 = AAU 2 = RMU 3 = ERU converted from AAU 4 = ERU converted from RMU 5 = CER 6 = tCER 7 = lCER
Supplement-ary Unit Type	3		Numeric	2	Blank for Kyoto-only Units, or as defined by STL
Unit Serial Block Start	4	All	Numeric	15	Unique numeric values assigned by registry from 1 – 999,999,999,999,999
Unit Serial Block End	5	All	Numeric	15	Unique numeric values assigned by registry from 1 – 999,999,999,999,999
Original Commitment Period	6	All	Numeric	2	1 – 99
Applicable Commitment Period	7	AAU, CER, ERU, lCER, tCER	Numeric	2	1 – 99
LULUCF Activity	8	RMU, CER, ERU, lCER, tCER	Numeric	3	1 = Afforestation and reforestation 2 = Deforestation 3 = Forest management 4 = Cropland management 5 = Grazing land management 6 = Revegetation

(cont.)

Figure F1: Serial Number Identifiers (cont.)

Identifier	Display Order	Identifier Required for the Following Unit Types	Data Type	Length	Range or Codes
Project Identifier	9	CER, ERU, lCER, tCER	Numeric	7	Unique numeric value assigned by registry for project
Track	10	ERU	Numeric	2	1 or 2
Expiry Date	11	lCER, tCER	Date		Expiry Date for lCERs or tCERs

5. Account Numbers

The account number shall be unique throughout all registries.

An account number for an account that is deactivated or deleted cannot be reused.

Account numbers are created and defined only by registries.

Holding accounts do not have an applicable commitment period.

Figure F2: Account Number Identifiers

Element	Display	Data Type	Length	Range or Codes
Originating Registry	1	Alphanumeric	3	Per country codes in ISO3166, as of 01 January, 2005, or "EB" for CDM projects approved by Executive Board
Account Type	2	Numeric	3	100 = Holding Account 110 = Pending Account 120 = Operator Holding Account 121 = Person Holding Account 210 = Net Source Cancellation Account 220 = Non-compliance Cancellation Account 230 = Voluntary Cancellation Account 240 = Excess Issuance Cancellation Account 300 = Retirement Account 410 = tCER Replacement Account for failure to submit certification request 421 = lCER Replacement Account for expiry 422 = lCER Replacement Account for reversal in carbon stage 423 = lCER Replacement Account for failure to submit certification report 500 = tCER Cancellation Account 600 = lCER Cancellation Account

(cont.)

Figure F2: Account Number Identifiers (cont.)

Element	Display	Data Type	Length	Range or Codes
Account Identifier	3	Numeric	15	Unique numeric values assigned by registry from 1 – 999,999,999,999,999
Applicable Commitment Period	4	Numeric	2	0 for all holding accounts 1 – 99 for retirement and cancellation accounts

6. Transaction Number

The transaction number shall be unique within the registries and within the ITL.

A transaction number for a transaction that is terminated or cancelled cannot be reused. Resubmission of a transfer for which a transaction has been terminated or cancelled shall be assigned a new, unique transaction number.

Transaction numbers are defined only by registries.

Figure F3: Transaction Number Identifiers

Identifier	Display	Data Type	Length	Range or Codes
Originating Registry	1	Alphanumeric	3	Per country codes in ISO3166, as of 01 January, 2005
Transaction Identifier	2	Numeric	15	Unique numeric values assigned by registry from 1 – 999,999,999,999,999

7. Reconciliation Numbers

The reconciliation number shall be unique throughout all registries and the ITL.

Reconciliation numbers are defined only by the ITL at the time that a reconciliation is requested by the ITL for a registry. The reconciliation ends when the ITL determines there are no discrepancies or when a manual intervention to correct inconsistencies is complete. If a resubmission of information is needed because of a data format problem, the same reconciliation number shall be used.

Figure F4: Reconciliation Number Identifiers

Identifier	Display Order	Data Type	Length	Range or Codes
Registry Identifier	1	Alphanumeric	3	Per country codes in ISO3166, as of 01 January, 2005 and "CDM" for CDM registry
Reconciliation Identifier	2	Numeric	15	Unique numeric values assigned by Transaction Log from 1 – 999,999,999,999,999

8. Project Numbers

The Project number shall be unique.

Project numbers are defined only by registries or the CDM Executive Board (in cooperation with the CDM Registry Administrator).

Figure F5: Project Identifiers

Identifier	Display Order	Data Type	Length	Range or Codes
Registry or Party Identifier	1	Alphanumeric	3	Per country codes in ISO3166 as of 01 January, 2005
Project Identifier	2	Numeric	7	Unique numeric values assigned by registry for project

Annex G
List of Codes

Annex G defines the codes for all support tables used in this technical specification. These codes define the acceptable values for elements defined in Annex F or otherwise required for data exchange.

Figure G1: List of Tables

Table	Description
Account Type Code	Account type categories.
Commitment Period Code	Identifies the current Commitment Period.
LULUCF Activity Code	Land Use, Land Use Change, and Forestry categories
Notification Type Code	Identifies the reason for the Notification.
Party Type	Defines the role of the Party.
Reconciliation Status	Identifies the status of a reconciliation process.
Transaction Status Code	Identifies the status of a transaction
Transaction Type Code	Identifies the type of transaction.
Unit Type Code	Identifies the type of unit.

Figure G2: 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)

Figure G3: 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 G4: LULUCF Activity Codes

Code	Description
1	Category 1: Afforestation and reforestation
2	Category 2: Deforestation
3	Category 3: Forest management
4	Category 4: Cropland management
5	Category 5: Grazing land management
6	Category 6: Revegetation

Figure G5: Notification Type Codes

Code	Description
1	Transaction Expiration (24 hours)
2	Carry-over Units
3	Failure to Submit Certification Report
4	Failure to Submit Certification Report - Reminder
5	Receipt of Certification Report
6	Reversal in Storage - Initial Notification
7	Reversal in Storage - Reminder
8	Impending Expiration of lCER
9	Impending Expiration of tCER
10	Failure to Submit Correction
11	Replacement for Reversal in Storage Met

Figure G6: Party Type Codes

Code	Description
1	Transferring Registry
2	Acquiring Registry

562 **Figure G7: Reconciliation Status Code**
563

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”

564
565
566 **Figure G8: Transaction Status Code**
567

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)"

568
569

Figure G9: Transaction Type Code

Code	Description
1	Issuance - Initial creation of a unit
2	Conversion - Transformation of a unit to create an ERU
3	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 a unit's validity
8	Expiry Date Change
10	Internal - Transfer of a unit/supplementary program transaction

Figure G10: 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 H
Test Protocols for Data Exchange Specification Implementation

This Annex will address the test requirements for verifying conformance with the Data Exchange Specifications version 1.0.

Annex I
Messaging Service Specification

1. Introduction

This annex provides information on the required XML message structures. Messages are passed in the registry system using the SOAP 1.1 protocol defined by the W3C [SOAP1.1], using the RPC/Encoded style.

2. SOAP Message Exchange

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML-based protocol that consists of three parts:

- An envelope that defines a framework for describing what is in a message and how to process it;
- A set of encoding rules for validation of defined elements;
- A convention for representing remote procedure calls and responses.

Requests (and responses) in the registry system are SOAP messages. Every SOAP message referred to in these specifications has a mandatory envelope and body element. The body contains the actual message to be delivered and processed.

3. Standards and Requirements

Due to the fact that the exchanges in the registry system will connect heterogeneous systems, the exchange of data must follow a common representation. Annex K (WSDL) describes the message structure and format of every message exchanged in the registry system. Any deviation from these named conventions shall result in the message being rejected.

3.1 Versioning

All messages shall comply with and be backward compatible with SOAP 1.1. The version of a SOAP message can be determined by the namespace defined for the SOAP message in the envelope. All SOAP messages will be transported by the HTTP1.1 protocol.

3.2 Encoding Styles

An encoding style is a set of rules that define exactly how data types are to be encoded into a common XML syntax. The SOAP1.1 messages in the registry system are defined using the RPC/Encoded encoding style, in order to guarantee the largest compatibility possible with existing tools and SOAP processors.

3.3 SOAP Interoperability

Although any implementation of SOAP must be designed to be platform- and language-independent, there are known interoperability problems between the different SOAP implementations. The WSDLs and SOAP messages submitted by the registries shall comply as much as possible with the Web Services Interoperability Organization (WS-I), Basic Profile 1.0a. This profile clarifies the grey areas of SOAP that are the root cause of interoperability problems. The WSDLs defined in Annex K are known to be compliant with the Web Service Interoperability Organization for all aspects of their definition.

3.4 SOAP-Specific Port

It is recommended that SOAP messages in the registry system use TCP well-known port 80 for HTTP and 443 for HTTPS. See [RFC 1700]. Because of its applicative nature, SOAP usage must be restricted to known hosts in the registry system and network administrators should take all necessary measures to allow incoming HTTP and HTTPS requests only from recognized hosts. Additionally, only outgoing HTTP and HTTPS requests to recognized hosts should be authorized.

4. SOAP Message Specifications

4.1 SOAP Envelope Element

SOAP messages are well-formed XML documents, the envelope must start with an envelope declaration which contains namespace declarations. The envelope element must be prefixed with an indicator of the namespace that defines the SOAP version that is applicable. The version is indicated by the namespace attribute, xmlns, included in the envelope element start tag. The namespace prefix could be any valid XML namespace string, but the convention usually adopted is as follows:

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

The namespace name SOAP-ENV is a symbol for http://schema.xmlsoap/soap/envelope. Although it may be referred to by any name, the URL part must be exactly as specified.

4.2 The Header Element

The Header element offers a framework for specifying additional application-level requirements. Due to various interpretations and incomplete support by SOAP processors and their WSDL compiling tools of the header information in SOAP messages, the registry system will not make use of the SOAP header functionality. Rather, information common to all messages will be embedded in the body of the messages.

4.3 The Body Element

The Body element is mandatory for all SOAP messages and contains information regarding the specific message being sent. Annex K defines the structure and layout of the messages.

4.3.1 Namespaces

The primary use of namespaces in XML documents is to enable identification of logical structures in documents by software modules such as query processors, stylesheet-driven rendering engines and schema-driven validators. Separate namespaces shall be used to distinguish elements that are associated with all types from those associated with operations. The following namespaces have been defined for the registry system:

- "urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0", the target namespace for all operations.
- "urn:KyotoProtocol:RegistrySystem:ITL:Types:1.0:0.0", the type namespace for all ITL-recognized elements and attributes.

Registries will use operations defined in the ITL namespaces (i.e. ITL for operations and ITL:Types for the data types), in order to reduce the possible overhead when transitioning from the STL-only operations to the ITL-STL cooperation.

5. WSDL Requirements

Web Services Description Language (WSDL) is a specification defining how a web service is described in a common XML grammar. It represents a contact between the service requestor and service provider which is platform and language independent. The WSDLs defined in Annex K will be made available from the STL public web site and/or from the ITL public web site.

Annex J
QA Checklist by Requirement

This Annex will list the requirements in the "Data Exchange Standards for Registry Systems under the Kyoto Protocol: Functional Specification, Draft version <7.0>" and cross references the sections of the report which address them.

Annex K
Description Language (WSDL) Documentation

1. Introduction

This annex contains the WSDL documents for data exchange for the Kyoto Protocol. The documents listed below specify how registries can communicate with the ITL and vice versa. These documents define the interfaces to the public web service methods on both the ITL and registries. These web service methods are the portals through which data can be passed into an application.

There are three WSDL documents listed below: Common, Registry, and Transaction Log. The Common WSDL describes the data types and messages used by the other two. The Registry WSDL describes the operations and the bindings that registries must support. The Registry WSDL imports the types and messages in the Common WSDL. The Transaction Log WSDL describes the operations and the bindings that the ITL will support. The Transaction Log WSDL also imports the types and messages in the Common WSDL.

In order to aid the understanding of non-technical persons, a summary of each operation is provided prior to the WSDL documents. WSDL documents are designed to be "read" by computers and not humans. The Web Service Method Summaries below reduce the numerous cross-references and repetitive nature of the WSDL documents to a more understandable form. The summaries are still detailed and reflect the complex nature of the data exchange system.

2. Summaries of Web Service Operations

2.1 acceptMessage

Figure K1: acceptMessage
Role(s): Registry and ITL

MessageRequest

from	string	
to	string	
majorVersion	int	
minorVersion	int	
messageContent	string	
messageDateTime	dateTime	

MessageResponse

resultIdentifier	int	
responseCodes	**ArrayOfInt**	Optional

2.2 acceptNotification

Figure K2: acceptNotification
Role(s): Registry and ITL

NotificationRequest

from	string		
to	string		
majorVersion	int		
minorVersion	int		
transactionIdentifier	string		
transactionStatus	int		
partyType	int		
evaluationResult	**ArrayOfEvaluationResult**		
	responseCode	int	
	unitBlockIdentifiers	**ArrayOfUnitBlockIdentifier**	Optional
		unitSerialBlockStart long	
		unitSerialBlockEnd long	
		originatingRegistryIdentifier string	

NotificationResponse

resultIdentifier	int	
responseCodes	**ArrayOfInt**	Optional

2.3 acceptProposal

Figure K3: acceptProposal
Role(s): Registry and ITL

ProposalRequest

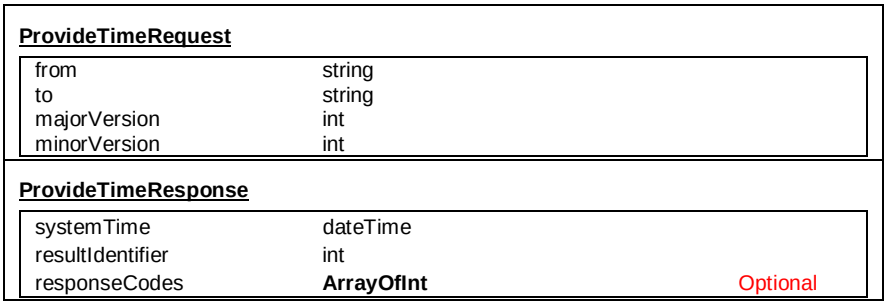
from	string		
to	string		
majorVersion	int		
minorVersion	int		
proposedTransaction	**ProposalTransaction**		
	transactionIdentifier	string	
	transactionType	int	
	suppTransactionType	Int	Optional
	transferringRegistryIdentifier	string	
	transferringRegistryAccountType	int	
	transferringRegistryAccountIdentifier	long	Optional
	acquiringRegistryIdentifier	string	Optional
	acquiringRegistryAccountType	int	
	acquiringRegistryAccountIdentifier	long	Optional
	proposalUnitBlocks		
	notificationIdentifier	Int	Optional
		ArrayOfTransactionUnitBlock	
		unitSerialBlockStart	long
		unitSerialBlockEnd	long
		originatingRegistryIdentifier	string
		unitType	int
		suppUnitType	int Optional
		originalCommitPeriod	int
		applicableCommitPeriod	int
		LULUCFActivity	int Optional
		projectIdentifier	int Optional
		track	int Optional
		blockRole	string Optional
		acquiringRegistryAccountType	int Optional
		acquiringRegistryAccountIdentifier	long Optional
		transferringRegistryAccountType	int Optional
		transferringRegistryAccountIdentifier	long Optional
		yearinCommitmentPeriod	int Optional
		installationIdentifier	long Optional
		expiryDate	dateTime Optional
		acquiringRegistryIdentifier	string Optional
		transferringRegistryIdentifier	string Optional

ProposalResponse

resultIdentifier	int	
responseCodes	**ArrayOfInt**	Optional

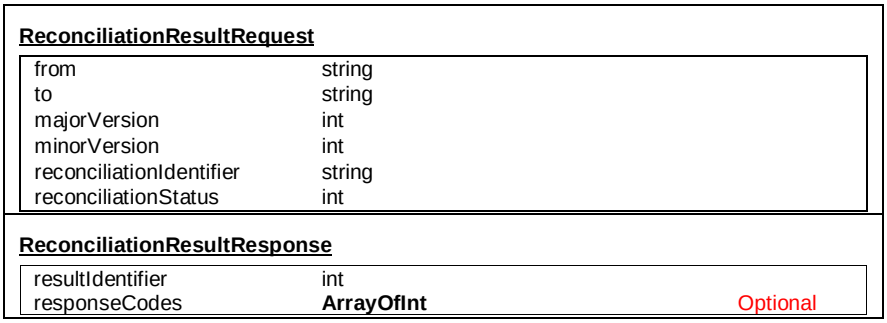
2.4 provideTime

Figure K4: provideTime
Role(s): Registry



2.5 receiveReconciliationResult

Figure K5: receiveReconciliationResult
Role(s): Registry, ITL



2.6 provideAuditTrail

Figure K6: provideAuditTrail
Role(s): Registry, ITL

ProvideAuditTrailRequest

from	string	
to	string	
majorVersion	int	
minorVersion	int	
reconciliationIdentifier	string	
auditTrailBeginDatetime	dateTime	
auditTrailEndDatetime	dateTime	
accountType	int	Optional
accountIdentifier	long	Optional
unitType	int	Optional
suppUnitType	int	Optional
unitBlockIdentifiers	**ArrayOfUnitBlockIdentifier**	Optional
	unitSerialBlockStart: long	
	unitSerialBlockEnd: long	
	originatingRegistryIdentifier: string	

ProvideAuditTrailResponse

resultIdentifier	int	
responseCodes	**ArrayOfInt**	Optional

2.7 provideTotals

Figure K7: provideTotals
Role(s): Registry, ITL

ProvideTotalsRequest

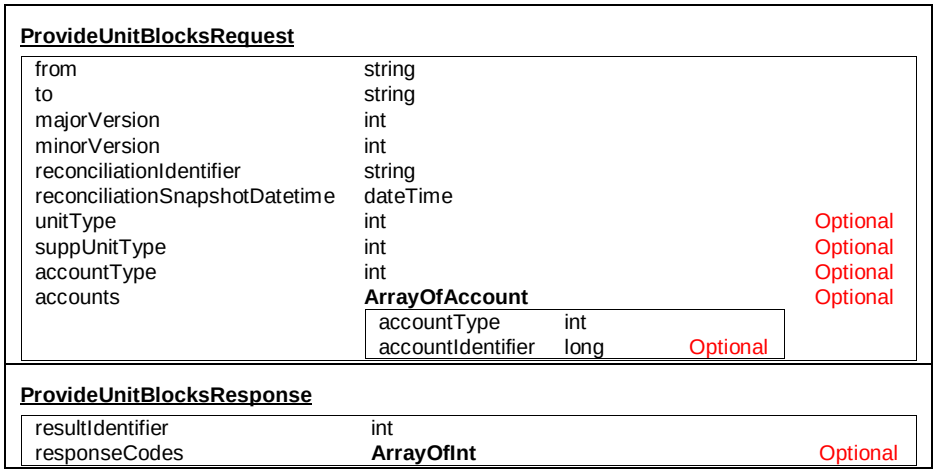
from	string	
to	string	
majorVersion	int	
minorVersion	int	
reconciliationIdentifier	string	
reconciliationSnapshotDatetime	dateTime	
reconciliationStatus	int	
unitType	int	Optional
suppUnitType	int	Optional
accountType	int	Optional
byAccountFlag	int	Optional

ProvideTotalsResponse

resultIdentifier	int	
responseCodes	**ArrayOfInt**	Optional

2.8 provideUnitBlocks

Figure K8: provideUnitBlocks
Role(s): Registry

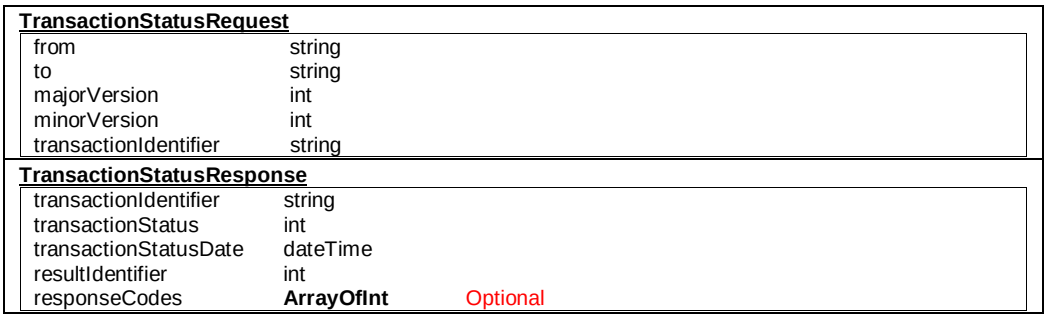


ProvideUnitBlocksRequest			
from	string		
to	string		
majorVersion	int		
minorVersion	int		
reconciliationIdentifier	string		
reconciliationSnapshotDatetime	dateTime		
unitType	int		Optional
suppUnitType	int		Optional
accountType	int		Optional
accounts	**ArrayOfAccount**		Optional
	accountType	int	
	accountIdentifier	long	Optional

ProvideUnitBlocksResponse		
resultIdentifier	int	
responseCodes	**ArrayOfInt**	Optional

2.9 getTransactionStatus

Figure K9: getTransactionStatus
Role(s): ITL



TransactionStatusRequest	
from	string
to	string
majorVersion	int
minorVersion	int
transactionIdentifier	string

TransactionStatusResponse		
transactionIdentifier	string	
transactionStatus	int	
transactionStatusDate	dateTime	
resultIdentifier	int	
responseCodes	**ArrayOfInt**	Optional

2.10 receiveTotals

Figure K10: receiveTotals
Role(s): ITL

ReceiveTotalsRequest

from	String	
to	string	
majorVersion	int	
minorVersion	int	
reconciliationIdentifier	string	
totals	**ArrayOfTotal**	
accountType	int	
accountIdentifier	long	Optional
unitType	int	
suppUnitType	int	Optional
unitCount	long	

ReceiveTotalsResponse

resultIdentifier	int	
responseCodes	**ArrayOfInt**	Optional

2.11 receiveUnitBlocks

Figure K11: receiveUnitBlocks
Role(s): ITL

ReceiveUnitBlocksRequest

from	string		
to	string		
majorVersion	int		
minorVersion	int		
reconciliationIdentifier	string		
unitBlocks	**ArrayOfUnitBlock**		
unitSerialBlockStart	long		
unitSerialBlockEnd	long		
originatingRegistryIdentifier	string		
unitType	int		Optional
suppUnitType	int		Optional
AccountType	int		
AccountIdentifier	long		Optional

ReceiveUnitBlocksResponse

resultIdentifier	int	
responseCode	**ArrayofInt**	Optional

2.12 receiveAuditTrail

Figure K12: receiveAuditTrail
Role(s): ITL

ReceiveAuditTrailRequest

from	string		
to	string		
majorVersion	int		
minorVersion	int		
reconciliationIdentifier	string		
transactions	**ArrayOfTransaction**		
	transactionIdentifier	string	
	transactionType	int	
	suppTransactionType	int	Optional
	transactionStatusDateTime	dateTime	
	transferringRegistryIdentifier	string	
	transferringRegistryAccountType	int	
	transferringRegistryAccountIdentifier	long	Optional
	acquiringRegistryIdentifier	string	
	acquiringRegistryAccountType	int	
	acquiringRegistryAccountIdentifier	long	Optional
	notificationIdentifier	int	Optional
	transactionBlocks	**ArrayOfTransactionUnitBlock**	

unitSerialBlockStart	long	
unitSerialBlockEnd	long	
originatingRegistryIdentifier	string	
unitType	int	Optional
suppUnitType	int	Optional
originalCommitPeriod	int	
applicableCommitPeriod	int	
LULUCFActivity	int	Optional
projectIdentifier	int	Optional
track	int	Optional
blockRole	string	Optional
transferringRegistryAccountType	int	Optional
transferringRegistryAccountIdentifier	long	Optional
acquiringRegistryAccountType	int	Optional
acquiringRegistryAccountIdentifier	long	Optional
yearinCommitmentPeriod	int	Optional
installationIdentifier	long	Optional
expiryDate	dateTime	Optional
acquiringRegistryIdentifier	string	Optional
transferringRegistryIdentifier	string	Optional

ReceiveAuditTrailResponse

resultIdentifier	int	
responseCodes	**ArrayOfInt**	Optional

2.13 accountManagement

[to be inserted]

3. Registry WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="RegistryService" targetNamespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"
xmlns:tns="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"
xmlns:typens="urn:KyotoProtocol:RegistrySystem:ITL:Types:1.0:0.0"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
        <import namespace="urn:KyotoProtocol:RegistrySystem:ITL:Types:1.0:0.0"
location="common.wsdl"/>
        <portType name="RegistryPort">
                <operation name="provideTime">
                        <input message="typens:provideTimeRequest"/>
                        <output message="typens:provideTimeResponse"/>
                </operation>
                <operation name="acceptMessage">
                        <input message="typens:acceptMessageRequest"/>
                        <output message="typens:acceptMessageResponse"/>
                </operation>
                <operation name="acceptProposal">
                        <input message="typens:acceptProposalRequest"/>
                        <output message="typens:acceptProposalResponse"/>
                </operation>
                <operation name="acceptNotification">
                        <input message="typens:acceptNotificationRequest"/>
                        <output message="typens:acceptNotificationResponse"/>
                </operation>
                <operation name="receiveReconciliationResult">
                        <input message="typens:receiveReconciliationResultRequest"/>
                        <output message="typens:receiveReconciliationResultResponse"/>
                </operation>
                <operation name="provideAuditTrail">
                        <input message="typens:provideAuditTrailRequest"/>
                        <output message="typens:provideAuditTrailResponse"/>
                </operation>
                <operation name="provideTotals">
                        <input message="typens:provideTotalsRequest"/>
                        <output message="typens:provideTotalsResponse"/>
                </operation>
                <operation name="provideUnitBlocks">
                        <input message="typens:provideUnitBlocksRequest"/>
                        <output message="typens:provideUnitBlocksResponse"/>
                </operation>
        </portType>
        <binding name="RegistryBinding" type="tns:RegistryPort">
                <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
                <operation name="provideTime">
                        <soap:operation soapAction="provideTime" style="rpc"/>
                        <input>
                                <soap:body use="encoded"
                                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                        </input>
                        <output>
                                <soap:body use="encoded"
                                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                        </output>
                </operation>
                <operation name="acceptMessage">
                        <soap:operation soapAction="acceptMessage" style="rpc"/>
                        <input>
                                <soap:body use="encoded"
                                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                        </input>
                        <output>
                                <soap:body use="encoded"
                                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                        </output>
                </operation>
```

```
        <operation name="acceptProposal">
                <soap:operation soapAction="acceptProposal" style="rpc"/>
                <input>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </input>
                <output>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </output>
        </operation>
        <operation name="acceptNotification">
                <soap:operation soapAction="acceptNotification" style="rpc"/>
                <input>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </input>
                <output>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>

                </output>
        </operation>
        <operation name="receiveReconciliationResult">
                <soap:operation soapAction="receiveReconciliationResult" style="rpc"/>
                <input>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </input>
                <output>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </output>
        </operation>
        <operation name="provideAuditTrail">
                <soap:operation soapAction="provideAuditTrail" style="rpc"/>
                <input>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </input>
                <output>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </output>
        </operation>
        <operation name="provideTotals">
                <soap:operation soapAction="provideTotals" style="rpc"/>
                <input>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </input>
                <output>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </output>
        </operation>
        <operation name="provideUnitBlocks">
                <soap:operation soapAction="provideUnitBlocks" style="rpc"/>
                <input>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
                                namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                        </input>
                        <output>
                                <soap:body use="encoded"
                                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                        </output>
                </operation>
        </binding>
        <service name="RegistryService">
                <port name="RegistryPort" binding="tns:RegistryBinding">
                        <soap:address location="http://REPLACE.WITH.ACTUAL.URL"/>
                </port>
        </service>
</definitions>
```

4. Transaction Log WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="TransactionLogService"
targetNamespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"
xmlns:tns="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"
xmlns:typens="urn:KyotoProtocol:RegistrySystem:ITL:Types:1.0:0.0"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
        <import namespace="urn:KyotoProtocol:RegistrySystem:ITL:Types:1.0:0.0"
location="common.wsdl"/>
        <portType name="TransactionLogPort">
                <operation name="acceptMessage">
                        <input message="typens:acceptMessageRequest"/>
                        <output message="typens:acceptMessageResponse"/>
                </operation>
                <operation name="acceptProposal">
                        <input message="typens:acceptProposalRequest"/>
                        <output message="typens:acceptProposalResponse"/>
                </operation>
                <operation name="acceptNotification">
                        <input message="typens:acceptNotificationRequest"/>
                        <output message="typens:acceptNotificationResponse"/>
                </operation>
                <operation name="getTransactionStatus">
                        <input message="typens:getTransactionStatusRequest"/>
                        <output message="typens:getTransactionStatusResponse"/>
                </operation>
                <operation name="receiveReconciliationResult">
                        <input message="typens:receiveReconciliationResultRequest"/>
                        <output message="typens:receiveReconciliationResultResponse"/>
                </operation>
                <operation name="provideAuditTrail">
                        <input message="typens:provideAuditTrailRequest"/>
                        <output message="typens:provideAuditTrailResponse"/>
                </operation>
                <operation name="provideTotals">
                        <input message="typens:provideTotalsRequest"/>
                        <output message="typens:provideTotalsResponse"/>
                </operation>
                <operation name="provideUnitBlocks">
                        <input message="typens:provideUnitBlocksRequest"/>
                        <output message="typens:provideUnitBlocksResponse"/>
                </operation>
                <operation name="receiveAuditTrail">
                        <input message="typens:receiveAuditTrailRequest"/>
                        <output message="typens:receiveAuditTrailResponse"/>
                </operation>
                <operation name="receiveTotals">
                        <input message="typens:receiveTotalsRequest"/>
                        <output message="typens:receiveTotalsResponse"/>
                </operation>
                <operation name="receiveUnitBlocks">
```

```
                <input message="typens:receiveUnitBlocksRequest"/>
                <output message="typens:receiveUnitBlocksResponse"/>
        </operation>
</portType>
<binding name="TransactionLogBinding" type="tns:TransactionLogPort">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
        <operation name="acceptMessage">
                <soap:operation soapAction="acceptMessage" style="rpc"/>
                <input>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </input>
                <output>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </output>
        </operation>
        <operation name="acceptProposal">
                <soap:operation soapAction="acceptProposal" style="rpc"/>
                <input>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </input>
                <output>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </output>
        </operation>
        <operation name="acceptNotification">
                <soap:operation soapAction="acceptNotification" style="rpc"/>
                <input>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </input>
                <output>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </output>
        </operation>
        <operation name="getTransactionStatus">
                <soap:operation soapAction="getTransactionStatus" style="rpc"/>
                <input>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </input>
                <output>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </output>
        </operation>
        <operation name="receiveReconciliationResult">
                <soap:operation soapAction="receiveReconciliationResult" style="rpc"/>
                <input>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </input>
                <output>
                        <soap:body use="encoded"
                         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                </output>
        </operation>
```

```
<operation name="provideAuditTrail">
        <soap:operation soapAction="provideAuditTrail" style="rpc"/>
        <input>
                <soap:body use="encoded"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
        </input>
        <output>
                <soap:body use="encoded"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
        </output>
</operation>
<operation name="provideTotals">
        <soap:operation soapAction="provideTotals" style="rpc"/>
        <input>
                <soap:body use="encoded"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
        </input>
        <output>
                <soap:body use="encoded"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
        </output>
</operation>
<operation name="provideUnitBlocks">
        <soap:operation soapAction="provideUnitBlocks" style="rpc"/>
        <input>
                <soap:body use="encoded"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
        </input>
        <output>
                <soap:body use="encoded"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
        </output>
</operation>
<operation name="receiveAuditTrail">
        <soap:operation soapAction="receiveAuditTrail" style="rpc"/>
        <input>
                <soap:body use="encoded"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
        </input>
        <output>
                <soap:body use="encoded"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
        </output>
</operation>
<operation name="receiveTotals">
        <soap:operation soapAction="receiveTotals" style="rpc"/>
        <input>
                <soap:body use="encoded"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
        </input>
        <output>
                <soap:body use="encoded"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
        </output>

</operation>
<operation name="receiveUnitBlocks">
        <soap:operation soapAction="receiveUnitBlocks" style="rpc"/>
        <input>
                <soap:body use="encoded"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
                                namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                        </input>
                        <output>
                                <soap:body use="encoded"
                                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                                 namespace="urn:KyotoProtocol:RegistrySystem:ITL:1.0:0.0"/>
                        </output>
                </operation>
        </binding>
        <service name="TransactionLogService">
                <port name="TransactionLogPort" binding="tns:TransactionLogBinding">
                        <soap:address location="http://REPLACE.WITH.ACTUAL.URL"/>
                </port>
        </service>
</definitions>
```

5. Common WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="RegistryService" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:typens="urn:KyotoProtocol:RegistrySystem:ITL:Types:1.0:0.0"
targetNamespace="urn:KyotoProtocol:RegistrySystem:ITL:Types:1.0:0.0"
xmlns="http://schemas.xmlsoap.org/wsdl/">
        <types>
                <schema targetNamespace="urn:KyotoProtocol:RegistrySystem:ITL:Types:1.0:0.0"
                        xmlns="http://www.w3.org/2001/XMLSchema">
                        <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
                        <complexType name="ArrayOfInt">
                                <complexContent>
                                        <restriction base="soapenc:Array">
                                                <attribute ref="soapenc:arrayType"
                                                        wsdl:arrayType="int[]"/>
                                        </restriction>
                                </complexContent>
                        </complexType>
                        <complexType name="Account">
                                <sequence>
                                        <element name="accountType" type="int"/>
                                        <element name="accountIdentifier" type="long" minOccurs="0"/>
                                </sequence>
                        </complexType>
                        <complexType name="ArrayOfAccount">
                                <complexContent>
                                        <restriction base="soapenc:Array">
                                                <attribute ref="soapenc:arrayType"
                                                        wsdl:arrayType="typens:Account[]"/>
                                        </restriction>
                                </complexContent>
                        </complexType>
                        <complexType name="Transaction">
                                <sequence>
                                        <element name="transactionIdentifier" type="string"/>
                                        <element name="transactionType" type="int"/>
                                        <element name="suppTransactionType" type="int" minOccurs="0"/>
                                        <element name="transactionStatusDateTime" type="dateTime"/>
                                        <element name="transferringRegistryIdentifier" type="string"/>
                                        <element name="transferringRegistryAccountType" type="int"/>
                                        <element name="transferringRegistryAccountIdentifier"
type="long"
                                                minOccurs="0"/>
                                        <element name="acquiringRegistryIdentifier" type="string"/>
                                        <element name="acquiringRegistryAccountType" type="int"/>
                                        <element name="acquiringRegistryAccountIdentifier" type="long"
                                                minOccurs="0"/>
                                        <element name="notificationIdentifier" type="long"
                                                minOccurs="0"/>
                                        <element name="transactionBlocks"
                                                type="typens:ArrayOfTransactionUnitBlock"/>
```

```
                        </sequence>
                    </complexType>
                    <complexType name="ArrayOfTransaction">
                        <complexContent>
                            <restriction base="soapenc:Array">
                                <attribute ref="soapenc:arrayType"
                                    wsdl:arrayType="typens:Transaction[]"/>
                            </restriction>
                        </complexContent>
                    </complexType>
                    <complexType name="ProposalTransaction">
                        <sequence>
                            <element name="transactionIdentifier" type="string"/>
                            <element name="transactionType" type="int"/>
                            <element name="suppTransactionType" type="int" minOccurs="0"/>
                            <element name="transferringRegistryIdentifier" type="string"/>
                            <element name="transferringRegistryAccountType" type="int"/>
                            <element name="transferringRegistryAccountIdentifier"
type="long"
                                minOccurs="0"/>
                            <element name="acquiringRegistryIdentifier" type="string"
                                minOccurs="0"/>
                            <element name="acquiringRegistryAccountType" type="int"/>
                            <element name="acquiringRegistryAccountIdentifier" type="long"
                                minOccurs="0"/>
                            <element name="notificationIdentifier" type="long"
                                minOccurs="0"/>
                            <element name="proposalUnitBlocks"
                                type="typens:ArrayOfTransactionUnitBlock"/>
                        </sequence>
                    </complexType>
                    <complexType name="UnitBlockIdentifier">
                        <sequence>
                            <element name="unitSerialBlockStart" type="long"/>
                            <element name="unitSerialBlockEnd" type="long"/>
                            <element name="originatingRegistryIdentifier" type="string"/>
                        </sequence>
                    </complexType>
                    <complexType name="ArrayOfUnitBlockIdentifier">
                        <complexContent>
                            <restriction base="soapenc:Array">
                                <attribute ref="soapenc:arrayType"
                                    wsdl:arrayType="typens:UnitBlockIdentifier[]"/>
                            </restriction>
                        </complexContent>
                    </complexType>
                    <complexType name="UnitBlock">
                        <sequence>
                            <element name="unitSerialBlockStart" type="long"/>
                            <element name="unitSerialBlockEnd" type="long"/>
                            <element name="originatingRegistryIdentifier" type="string"/>
                            <element name="unitType" type="int" minOccurs="0"/>
                            <element name="suppUnitType" type="int" minOccurs="0"/>
                            <element name="accountType" type="int"/>
                            <element name="accountIdentifier" type="long" minOccurs="0"/>
                        </sequence>
                    </complexType>
                    <complexType name="ArrayOfUnitBlock">
                        <complexContent>
                            <restriction base="soapenc:Array">
                                <attribute ref="soapenc:arrayType"
                                    wsdl:arrayType="typens:UnitBlock[]"/>
                            </restriction>
                        </complexContent>
                    </complexType>
                    <complexType name="TransactionUnitBlock">
                        <sequence>
                            <element name="unitSerialBlockStart" type="long"/>
                            <element name="unitSerialBlockEnd" type="long"/>
                            <element name="originatingRegistryIdentifier" type="string"/>
                            <element name="unitType" type="int" minOccurs="0"/>
```

```
                                        <element name="suppUnitType" type="int" minOccurs="0"/>
                                        <element name="originalCommitPeriod" type="int"/>
                                        <element name="applicableCommitPeriod" type="int"/>
                                        <element name="LULUCFActivity" type="int" minOccurs="0"/>
                                        <element name="projectIdentifier" type="int" minOccurs="0"/>
                                        <element name="track" type="int" minOccurs="0"/>
                                        <element name="blockRole" type="string" minOccurs="0"/>
                                        <element name="transferringRegistry" type="string"
                                                minOccurs="0"/>
                                        <element name="transferringRegistryAccountType" type="int"
                                                minOccurs="0"/>
                                        <element name="transferringRegistryAccountIdentifier"
type="long"
                                                minOccurs="0"/>
                                        <element name="acquiringRegistry" type="string" minOccurs="0"/>
                                        <element name="acquiringRegistryAccountType" type="int"
                                                minOccurs="0"/>
                                        <element name="acquiringRegistryAccountIdentifier" type="long"
                                                minOccurs="0"/>
                                        <element name="yearInCommitmentPeriod" type="int"
minOccurs="0"/>
                                        <element name="installationIdentifier" type="long"
                                                minOccurs="0"/>
                                        <element name="expiryDate" type="dateTime" minOccurs="0"/>
                                </sequence>
                        </complexType>
                        <complexType name="ArrayOfTransactionUnitBlock">
                                <complexContent>
                                        <restriction base="soapenc:Array">
                                                <attribute ref="soapenc:arrayType"

                                                wsdl:arrayType="typens:TransactionUnitBlock[]"/>
                                        </restriction>
                                </complexContent>
                        </complexType>
                        <complexType name="Total">
                                <sequence>
                                        <element name="accountType" type="int"/>
                                        <element name="accountIdentifier" type="long" minOccurs="0"/>
                                        <element name="unitType" type="int"/>
                                        <element name="suppUnitType" type="int" minOccurs="0"/>
                                        <element name="unitCount" type="long"/>
                                </sequence>
                        </complexType>
                        <complexType name="ArrayOfTotal">
                                <complexContent>
                                        <restriction base="soapenc:Array">
                                                <attribute ref="soapenc:arrayType"
                                                        wsdl:arrayType="typens:Total[]"/>
                                        </restriction>
                                </complexContent>
                        </complexType>
                        <complexType name="EvaluationResult">
                                <sequence>
                                        <element name="responseCode" type="int"/>
                                        <element name="unitBlockIdentifiers"
                                                type="typens:ArrayOfUnitBlockIdentifier"
                                        minOccurs="0"/>
                                </sequence>
                        </complexType>
                        <complexType name="ArrayOfEvaluationResult">
                                <complexContent>
                                        <restriction base="soapenc:Array">
                                                <attribute ref="soapenc:arrayType"
                                                        wsdl:arrayType="typens:EvaluationResult[]"/>
                                        </restriction>
                                </complexContent>
                        </complexType>
                        <complexType name="ProvideTimeRequest">
                                <sequence>
                                        <element name="from" type="string"/>
```

```
                <element name="to" type="string"/>
                <element name="majorVersion" type="int"/>
                <element name="minorVersion" type="int"/>
        </sequence>
</complexType>
<complexType name="ProvideTimeResponse">
        <sequence>
                <element name="systemTime" type="dateTime"/>
                <element name="resultIdentifier" type="int"/>
                <element name="responseCodes" type="typens:ArrayOfInt"
                        minOccurs="0"/>
        </sequence>
</complexType>
<complexType name="MessageRequest">
        <sequence>
                <element name="from" type="string"/>
                <element name="to" type="string"/>
                <element name="majorVersion" type="int"/>
                <element name="minorVersion" type="int"/>
                <element name="messageContent" type="string"/>
                <element name="messageDateTime" type="dateTime"/>
        </sequence>
</complexType>
<complexType name="MessageResponse">
        <sequence>
                <element name="resultIdentifier" type="int"/>
                <element name="responseCodes" type="typens:ArrayOfInt"
                        minOccurs="0"/>
        </sequence>
</complexType>
<complexType name="ProposalRequest">
        <sequence>
                <element name="from" type="string"/>
                <element name="to" type="string"/>
                <element name="majorVersion" type="int"/>
                <element name="minorVersion" type="int"/>
                <element name="proposedTransaction"
                        type="typens:ProposalTransaction"/>
        </sequence>
</complexType>
<complexType name="ProposalResponse">
        <sequence>
                <element name="resultIdentifier" type="int"/>
                <element name="responseCodes" type="typens:ArrayOfInt"
                        minOccurs="0"/>
        </sequence>
</complexType>
<complexType name="NotificationRequest">
        <sequence>
                <element name="from" type="string"/>
                <element name="to" type="string"/>
                <element name="majorVersion" type="int"/>
                <element name="minorVersion" type="int"/>
                <element name="transactionIdentifier" type="string"/>
                <element name="transactionStatus" type="int"/>
                <element name="partyType" type="int"/>
                <element name="evaluationResult"
                        type="typens:ArrayOfEvaluationResult"/>
        </sequence>
</complexType>
<complexType name="NotificationResponse">
        <sequence>
                <element name="resultIdentifier" type="int"/>
                <element name="responseCodes" type="typens:ArrayOfInt"
                        minOccurs="0"/>
        </sequence>
</complexType>
<complexType name="ReconciliationResultRequest">
        <sequence>
                <element name="from" type="string"/>
                <element name="to" type="string"/>
```

```
                                        <element name="majorVersion" type="int"/>
                                        <element name="minorVersion" type="int"/>
                                        <element name="reconciliationIdentifier" type="string"/>
                                        <element name="reconciliationStatus" type="int"/>
                                </sequence>
                        </complexType>
                        <complexType name="ReconciliationResultResponse">
                                <sequence>
                                        <element name="resultIdentifier" type="int"/>
                                        <element name="responseCodes" type="typens:ArrayOfInt"
                                                minOccurs="0"/>
                                </sequence>
                        </complexType>
                        <complexType name="ProvideUnitBlocksRequest">
                                <sequence>
                                        <element name="from" type="string"/>
                                        <element name="to" type="string"/>
                                        <element name="majorVersion" type="int"/>
                                        <element name="minorVersion" type="int"/>
                                        <element name="reconciliationIdentifier" type="string"/>
                                        <element name="reconciliationSnapshotDatetime"
type="dateTime"/>
                                        <element name="unitType" type="int" minOccurs="0"/>
                                        <element name="suppUnitType" type="int" minOccurs="0"/>
                                        <element name="accountType" type="int" minOccurs="0"/>
                                        <element name="accounts" type="typens:ArrayOfAccount"
                                                minOccurs="0"/>
                                </sequence>
                        </complexType>
                        <complexType name="ProvideUnitBlocksResponse">
                                <sequence>
                                        <element name="resultIdentifier" type="int"/>
                                        <element name="responseCodes" type="typens:ArrayOfInt"
                                                minOccurs="0"/>
                                </sequence>
                        </complexType>
                        <complexType name="ProvideAuditTrailRequest">
                                <sequence>
                                        <element name="from" type="string"/>
                                        <element name="to" type="string"/>
                                        <element name="majorVersion" type="int"/>
                                        <element name="minorVersion" type="int"/>
                                        <element name="reconciliationIdentifier" type="string"/>
                                        <element name="auditTrailBeginDatetime" type="dateTime"/>
                                        <element name="auditTrailEndDatetime" type="dateTime"/>
                                        <element name="accountType" type="int" minOccurs="0"/>
                                        <element name="accountIdentifier" type="long" minOccurs="0"/>
                                        <element name="unitType" type="int" minOccurs="0"/>
                                        <element name="suppUnitType" type="int" minOccurs="0"/>
                                        <element name="unitBlockIdentifiers"
                                                type="typens:ArrayOfUnitBlockIdentifier"
                                        minOccurs="0"/>
                                </sequence>
                        </complexType>
                        <complexType name="ProvideAuditTrailResponse">
                                <sequence>
                                        <element name="resultIdentifier" type="int"/>
                                        <element name="responseCodes" type="typens:ArrayOfInt"
                                                minOccurs="0"/>
                                </sequence>
                        </complexType>
                        <complexType name="ProvideTotalsRequest">
                                <sequence>
                                        <element name="from" type="string"/>
                                        <element name="to" type="string"/>
                                        <element name="majorVersion" type="int"/>
                                        <element name="minorVersion" type="int"/>
                                        <element name="reconciliationIdentifier" type="string"/>
                                        <element name="reconciliationSnapshotDatetime"
type="dateTime"/>
                                        <element name="reconciliationStatus" type="int"/>
```

```
                        <element name="unitType" type="int" minOccurs="0"/>
                        <element name="suppUnitType" type="int" minOccurs="0"/>
                        <element name="accountType" type="int" minOccurs="0"/>
                        <element name="byAccountFlag" type="int" minOccurs="0"/>
                </sequence>
        </complexType>
        <complexType name="ProvideTotalsResponse">
                <sequence>
                        <element name="resultIdentifier" type="int"/>
                        <element name="responseCodes" type="typens:ArrayOfInt"
                                minOccurs="0"/>
                </sequence>
        </complexType>
        <complexType name="ReceiveTotalsRequest">
                <sequence>
                        <element name="from" type="string"/>
                        <element name="to" type="string"/>
                        <element name="majorVersion" type="int"/>
                        <element name="minorVersion" type="int"/>
                        <element name="reconciliationIdentifier" type="string"/>
                        <element name="totals" type="typens:ArrayOfTotal"/>
                </sequence>
        </complexType>
        <complexType name="ReceiveTotalsResponse">
                <sequence>
                        <element name="resultIdentifier" type="int"/>
                        <element name="responseCodes" type="typens:ArrayOfInt"
                                minOccurs="0"/>
                </sequence>
        </complexType>
        <complexType name="TransactionStatusRequest">
                <sequence>
                        <element name="from" type="string"/>
                        <element name="to" type="string"/>
                        <element name="majorVersion" type="int"/>
                        <element name="minorVersion" type="int"/>
                        <element name="transactionIdentifier" type="string"/>
                </sequence>
        </complexType>
        <complexType name="TransactionStatusResponse">
                <sequence>
                        <element name="transactionIdentifier" type="string"/>
                        <element name="transactionStatus" type="int"/>
                        <element name="transactionStatusDateTime" type="dateTime"/>
                        <element name="resultIdentifier" type="int"/>
                        <element name="responseCodes" type="typens:ArrayOfInt"
                                minOccurs="0"/>
                </sequence>
        </complexType>
        <complexType name="ReceiveAuditTrailRequest">
                <sequence>
                        <element name="from" type="string"/>
                        <element name="to" type="string"/>
                        <element name="majorVersion" type="int"/>
                        <element name="minorVersion" type="int"/>
                        <element name="reconciliationIdentifier" type="string"/>
                        <element name="transactions" type="typens:ArrayOfTransaction"/>
                </sequence>
        </complexType>
        <complexType name="ReceiveAuditTrailResponse">
                <sequence>
                        <element name="resultIdentifier" type="int"/>
                        <element name="responseCodes" type="typens:ArrayOfInt"
                                minOccurs="0"/>
                </sequence>
        </complexType>
        <complexType name="ReceiveUnitBlocksRequest">
                <sequence>
                        <element name="from" type="string"/>
                        <element name="to" type="string"/>
                        <element name="majorVersion" type="int"/>
```

```
                                <element name="minorVersion" type="int"/>
                                <element name="reconciliationIdentifier" type="string"/>
                                <element name="unitBlocks" type="typens:ArrayOfUnitBlock"/>
                        </sequence>
                </complexType>
                <complexType name="ReceiveUnitBlocksResponse">
                        <sequence>
                                <element name="resultIdentifier" type="int"/>
                                <element name="responseCodes" type="typens:ArrayOfInt"
                                        minOccurs="0"/>
                        </sequence>
                </complexType>
        </schema>
</types>
<message name="provideTimeRequest">
        <part name="provideTimeRequest" type="typens:ProvideTimeRequest"/>
</message>
<message name="provideTimeResponse">
        <part name="provideTimeResponse" type="typens:ProvideTimeResponse"/>
</message>
<message name="acceptMessageRequest">
        <part name="acceptMessageRequest" type="typens:MessageRequest"/>
</message>
<message name="acceptMessageResponse">
        <part name="acceptMessageResponse" type="typens:MessageResponse"/>
</message>
<message name="acceptProposalRequest">
        <part name="acceptProposalRequest" type="typens:ProposalRequest"/>
</message>
<message name="acceptProposalResponse">
        <part name="acceptProposalResponse" type="typens:ProposalResponse"/>
</message>
<message name="acceptNotificationRequest">
        <part name="acceptNotificationRequest" type="typens:NotificationRequest"/>
</message>
<message name="acceptNotificationResponse">
        <part name="acceptNotificationResponse" type="typens:NotificationResponse"/>
</message>
<message name="receiveReconciliationResultRequest">
        <part name="receiveReconciliationResultRequest"
                type="typens:ReconciliationResultRequest"/>
</message>
<message name="receiveReconciliationResultResponse">
        <part name="receiveReconciliationResultResponse"
                type="typens:ReconciliationResultResponse"/>
</message>
<message name="provideAuditTrailRequest">
        <part name="provideAuditTrailRequest" type="typens:ProvideAuditTrailRequest"/>
</message>
<message name="provideAuditTrailResponse">
        <part name="provideAuditTrailResponse" type="typens:ProvideAuditTrailResponse"/>
</message>
<message name="provideTotalsRequest">
        <part name="provideTotalsRequest" type="typens:ProvideTotalsRequest"/>
</message>
<message name="provideTotalsResponse">
        <part name="provideTotalsResponse" type="typens:ProvideTotalsResponse"/>
</message>
<message name="provideUnitBlocksRequest">
        <part name="provideUnitBlocksRequest" type="typens:ProvideUnitBlocksRequest"/>
</message>
<message name="provideUnitBlocksResponse">
        <part name="provideUnitBlocksResponse" type="typens:ProvideUnitBlocksResponse"/>
</message>
<message name="getTransactionStatusRequest">
        <part name="getTransactionStatusRequest" type="typens:TransactionStatusRequest"/>
</message>
<message name="getTransactionStatusResponse">
        <part name="getTransactionStatusResponse" type="typens:TransactionStatusResponse"/>
</message>
<message name="receiveAuditTrailRequest">
```

```
                <part name="receiveAuditTrailRequest" type="typens:ReceiveAuditTrailRequest"/>
        </message>
        <message name="receiveAuditTrailResponse">
                <part name="receiveAuditTrailResponse" type="typens:ReceiveAuditTrailResponse"/>
        </message>
        <message name="receiveTotalsRequest">
                <part name="receiveTotalsRequest" type="typens:ReceiveTotalsRequest"/>
        </message>
        <message name="receiveTotalsResponse">
                <part name="receiveTotalsResponse" type="typens:ReceiveTotalsResponse"/>
        </message>
        <message name="receiveUnitBlocksRequest">
                <part name="receiveUnitBlocksRequest" type="typens:ReceiveUnitBlocksRequest"/>
        </message>
        <message name="receiveUnitBlocksResponse">
                <part name="receiveUnitBlocksResponse" type="typens:ReceiveUnitBlocksResponse"/>
        </message>
</definitions>
```

6. Account Management WSDL

[to be inserted]