Framework Convention on
Climate Change

Communication standards for interoperability and recommended practices for transactions with internationally transferred mitigation outcomes

Version 1.0

Contents

Abbreviations and acronyms

AEF	Agreed electronic format
AES	Advanced Encryption Standard
API	Application Programming Interface
CARP	Centralized Accounting and Reporting Platform
CMA	Conference of the Parties serving as the meeting of the Parties to the Paris Agreement
CSV	Comma-Separated Values
GNU	GNU's not Unix
GZIP	GNU zip
HTTP	Hypertext transfer protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISO	International Organization for Standardization
ITMO	Internationally Transferred Mitigation Outcomes
JPG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
MB	Megabyte
MIME	Multipurpose Internet Mail Extensions
NDC	Nationally Determined Contribution
PDF	Portable Document Format
PNG	Portable Network Graphics
POSIX	Portable Operating System Interface
REST	Representational State Transfer
RFC	Request for Comments
RSA	Registry Systems Administrators
SHA	Secure Hash Algorithms
TAR	Tape Archive
TLS	Transport Layer Security
UML	Unified Modeling Language
UTC	Coordinated Universal Time

I. Introduction

A. Mandate

1. Decision 6/CMA.4, paragraph 32, requests the secretariat to develop, publish and periodically update, for participating Parties opting to apply the guidance referred to in annex I, chapter I.B, standards and recommended practices for electronic recording of data and information related to internationally transferred mitigation outcomes, and communication standards for interoperability and transactions with ITMOs, including record-keeping arrangements, data security protocols, risk management and disaster recovery procedures, and other practices, as necessary, including with inputs from the RSA forum.

2. This document contains the communication standards for interoperability and transactions with ITMOs (hereunder "these standards").

B. Applicability

3. These standards are intended for use by Parties utilizing interoperable registries for ITMO transfer and tracking.

4. While primarily aimed at the abovementioned Parties, these standards may also prove beneficial, in whole or in part, to other Parties and non-Parties.

C. Objectives

5. The guidance in decision 6/CMA.4, annex I, chapter I.B describes two primary objectives for these standards:

(a) **Mitigate data consistency risks**: "Implement appropriate standards and procedures to mitigate risks to the consistency of data, including through communication of data about the transfer and reconciliation procedures within and between registries";

(b) **Ensure non-repudiation**: "Interoperability of registries should be implemented in such a way that neither Party to an inter-registry transfer could later repudiate the existence, type, time or content of the transfer".

D. Scope

6. These standards include:

(a) **Section I** – Introduction: this section outlines the mandate and the related scope of these standards;

(b) **Section II** – Guiding principles: this section outlines the general principles and approach that is taken by these standards, ensuring robust accounting, prevention of double counting, business validation flexibility, timeliness, security, and streamlined operations;

(c) **Section III –** Processes: this section describes the processes related to the management and exchange of ITMOs;

(d) **Section IV –** Technical requirements: this section specifies the timing, security measures, data retention and related non-functional requirements of the registry systems;

(e) **Annex I –** Attributes of ITMOs: this annex contains the descriptions of all attributes of ITMOs;

(f) **Annex II** – Attributes of transactions: this annex contains the descriptions of all attributes of transaction;

(g) **Annex III** – Transaction validations: this annex lists the validations applicable to transactions;

(h) **Annex IV –** Account Type: this annex lists the types of accounts that registries may use to hold and transfer ITMOs;

(i) **Annex V –** Definitions: this annex provides definitions relevant to these standards.

E. Derived documents

7. The API documentation is based on these standards.

8. The operational procedures, listed in table 1 below, are also based on these standards.

Table 1
Derived operational procedures

Operational procedure	Purpose	Paragraph
Change management	To detail the process related to change management and versioning.	IV.H.148
Contact management	To assure that registry administrators are identified, and related contact details are kept up-to-date.	III.G.3.98
Correction of transaction	To document how incomplete or erroneous transactions are resolved.	III.B.7.65
Data retention	To detail the practices related to data retention and preservation.	III.H.2.113
Disaster recovery	To ensure business continuity of the service.	A.1
Operational status	To detail the process of managing the operational status of a registry.	III.G.4.103
Sandbox testing environment	To detail the utilization of the sandbox testing environment.	IV.D.137
Problem management	To describe how identified technical problems are managed within registry systems.	I.H
Reconciliation	To detail a standard approach to automatic reconciliation of records between registry systems.	0 III.F.4.95
Release management	To coordinate release of changes throughout registry systems.	I.H.15
Reversal of transaction	To document pre-requisites, steps, roles, and responsibilities involved in undoing a previously completed transaction between registry systems.	III.B.7.65
Risk management	To identify, assess, and prioritize risks to ensure they are understood and managed proactively.	I.A.1
Security	To specify a standardized response strategy in the event of a declared information security related incident on one or more registry systems.	IV.C.132
Testing	To document how coordinated tests are to be performed before a registry can interact with other production registries.	IV.D.137

F. Key words

9. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this

document are to be interpreted as described in IETF RFC 2119, and apply to Parties opting to follow these standards.

10. Within the context of this document, the term "registration" refers to the registration of units (authorized mitigation outcomes) by the originating Party registry with an interoperability hub, via an API call, as described in §III.A.24(b) and §III.B.2.

G. Graphical notation for figures

11. Unless otherwise specified, UML is used in figures of these standards as a graphical notation for representing the structure, behaviour, and interactions of the registry system's components.

H. Future collaborative development

12. Registry administrators are RECOMMENDED to promptly raise issues if they identify discrepancies, inconsistencies, inefficiencies, or any other problems with these standards.

13. Registry administrators SHOULD also suggest any enhancements they deem beneficial.

14. Registry administrators who are in a position to do so SHOULD actively participate in the resolution of the raised issues, as well as in the implementation of the needed improvements in these standards.

15. To facilitate the process of identifying, discussing, suggesting and integrating improvements, a collaborative environment is REQUIRED to be established. Within this environment, these standards SHALL be continuously developed, ensuring tracking of any changes and clear identification of future releases.

II. Guiding principles

16. These standards have been elaborated following these guiding principles:

(a) Robust accounting: information concerning ITMO transactions are documented and recorded accurately, and ITMO holdings, when reported, reflect the true position of a registry;

(b) Avoidance of double counting: ITMOs are only located in one account at any given time and maintain strict consistency between balances of ITMOs in each account and their ledger. Real-time consistency checks are performed at the time of transaction proposal, frequent reconciliations are undertaken, and in the event of an inconsistency, a predictable process for resolving it is in place;

(c) Business validations: it is up to registries to accept or reject transaction proposals, providing Parties with the flexibility to comply with their domestic rules and regulations;

(d) Timeliness: all transactions and all operations adhere to strict timeframes for finalization; information can be reported promptly, and cut-off date and times are defined for the purpose of reporting;

(e) Security: communications between registries are confidential, fault-resilient, reliable, and non-repudiable. A registry cannot access another Party's transactions, notifications or any other data. No confidential information is exchanged beyond what is strictly necessary to process transactions, enable non-repudiation of transactions, and traceability of ITMOs;

(f) Self-service: registry initialization and testing operations are streamlined by providing automated self-service options. For example, planned registry testing and data seeding, can reduce operational costs and support delivery times;

(g) Low barrier of entry: the API supporting these standards enable registries to interoperate without the need to implement a server. A polling architecture dispenses with the need for registries to implement a server-side API, and its associated infrastructure (e.g. http servers, 2-way SSL). This is demonstrated in the transaction flows in para. 27.
The polling architecture allows registries to be intermittently offline without endangering their tracking of ITMOs, lowering the service levels required and allowing for registries that would only perform few transactions to effectively interoperate without losing the benefits listed above;

(h) Pull and view: the API supporting these standards enables registries to pull and view data and information on holdings and transaction history, consistent with para. 49 of decision 4/CMA.6.

17. These standards anticipate that interoperability between registries, that opt to apply these standards, is facilitated by an interoperability hub. The options by which these standards may be applied, each of which incorporate an interoperability hub, are depicted in Figure 1 (note that Figure 1 is purely a logical depiction, other than each subcomponent requiring an IP address, there is no other implication regarding their implementation):

(a) Option 1 – Independently provided interoperability hub

Three Parties[1], owning Registry 1, Registry 2, and Registry 3 respectively, wish to interoperate. They co-develop or procure an interoperability hub, independent from any of the Parties, that implement these standards.

(b) Option 2 – Independently provided consolidated registry system

[1] The number of Parties could be any number, but three is used here for illustrative purposes.

Three Parties wish to interoperate. They do not currently own individual registries. They co-develop or acquire a consolidated registry system providing each Party a secure, independent, logically segregated registry section, Registry section 1, Registry section 2, Registry section 3, that interoperate with each other according to these standards, via an interoperability hub subcomponent.

(c) Option 3 – Party provided externally interoperable registry system

Three Parties wish to interoperate. Party 1 develops or acquires a registry system that includes an externally interoperable hub subcomponent, and registry sections for Party registries. Party 1 uses a registry section as its own registry, and interoperates with the other Parties, that own Registry 2 and Registry 3 respectively.

(d) Option 4 – Independently provided consolidated registry system with external interoperability.

An independent organisation develops or acquires a registry system that includes an externally interoperable hub subcomponent, and registry sections for Party registries. Some Parties use secure, independent, logically segregated registry sections, other Parties use externally connected registries for interoperability.

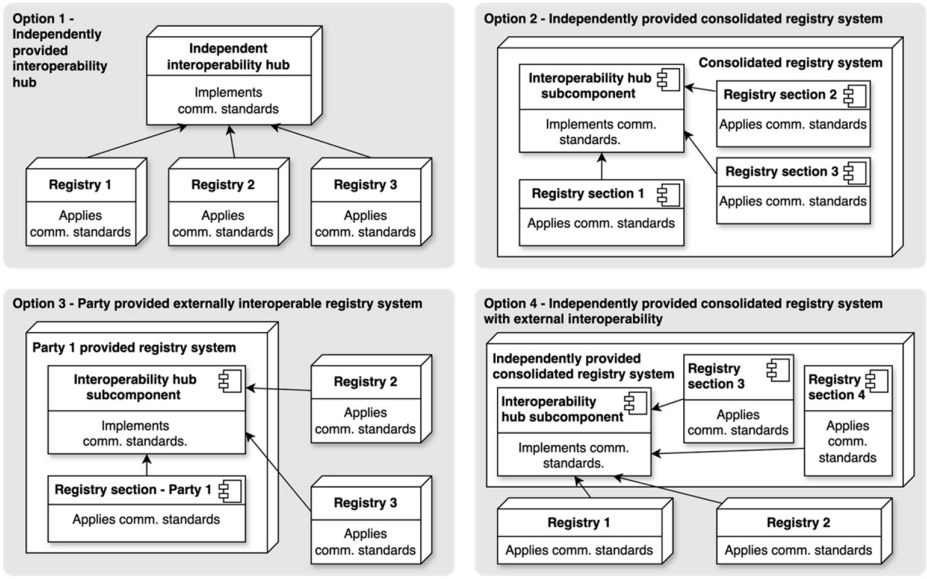

Figure 1 - Options for applying these interoperability standards

18. Each participating Party MUST submit, on an annual basis, information in an agreed electronic format (AEF) pursuant to annex, chapter IV.B of decision 2/CMA.3 through the submission portal of the CARP according to annex I, chapter II.A of decision 6/CMA.4 (as depicted in Figure 2). The submission portal SHALL have a human- and machine-readable interface for uploading information, including for recording data in the Article 6 database pursuant to decision 6/CMA.4, annex I, chapter II.A.

19. The secretariat SHALL apply these standards to the international registry referred to in decision 2/CMA.3, annex, chapter VI.A. In implementing these standards, the secretariat:

(a) Provides registry sections for Parties that do not have, or have access to a registry and/or that do not wish to establish and maintain their own registry. The international registry, including each of its registry sections, adhere to the consolidated system requirements set out in section IV.E;

(b) Operates an interoperability hub to facilitate communications between registries, including with the Article 6.4 mechanism registry referred to in decision 3/CMA.3, annex, para. 63, for Parties that choose not to establish and maintain a separate interoperability hub;

(c) Designates the international registry administrator as the interoperability hub administrator under these standards, with the secretariat fulfilling that role.

20. Parties who opt for applying the guidance in annex I, chapter I.B of decision 6/CMA.4 are REQUIRED to interoperate in the manner prescribed by these standards (registries 1, 2, 3, 8, and 9 in Figure 2).

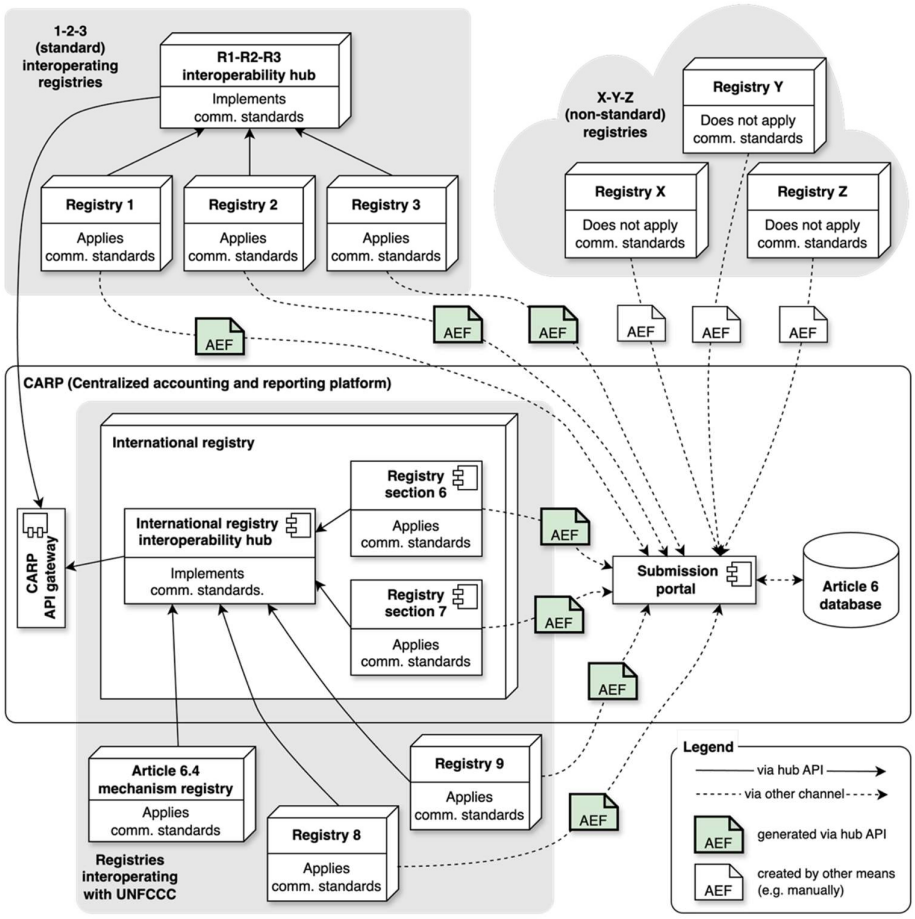

Figure 2 - Overview of different registry arrangements

21. Figure 2 depicts how different registries apply, or do not apply, these standards, and interact with UNFCCC secretariat:

(a) Registries 1, 2, and 3 use Option (1) from Figure 1 above. They develop or acquire an interoperability hub that implements these standards. The interoperability hub accesses the CARP API gateway, and the Parties submit AEF reports through the CARP submission portal.

Note that even though Registries 1, 2, and 3 implement these standards, they are not interoperating with the international registry, nor are they interoperating with the Article 6.4 mechanism registry, as they are using their own interoperability hub (R1-R2-R3 interoperability hub).

(b) The UNFCCC secretariat provides the international registry that includes registry sections (registry sections 6 and 7 in Figure 2), and an interoperability hub as subcomponents. The registry sections and interoperability hub apply

and implement these standards. The Article 6.4 mechanism registry SHALL apply these standards (as shown in Figure 2), and SHALL be connected to the international registry's interoperability hub. External registries that opt to interoperate via the international registry's interoperability hub (registries 8 and 9 in Figure 2) are REQUIRED to apply these standards.

This is an instantiation of Option (4) of Figure 1.

As the Article 6.4 mechanism registry SHALL be connected to the international registry's interoperability hub, other registries that require interoperability with the Article 6.4 mechanism registry will need to be connected to the international registry's interoperability hub, and are REQUIRED to apply these standards.

(c) Parties that opt no to apply the guidance of decision 6/CMA.4 MAY benefit from applying certain provisions of these standards but SHALL NOT be connected to the international registry (e.g. registries X, Y, and Z in Figure 2).;

III. Processes

A. The scope of these standards in the lifecycle of Mitigation Outcomes and ITMOs

22. Decision 6/CMA.4, paragraph 32, requires these standards to apply to transactions of ITMOs, whereas decision 2/CMA.3, annex, paragraph 29 requires registries to track authorizations and first transfers.

Consequently, these standards apply from when an authorized MO is registered with the interoperability hub, and continue to apply after the ITMO is used or cancelled, as there are restrictions on used or cancelled ITMOs.

For the purpose of these standards, the acronym MO is used, when necessary, to refer to such (pre-)ITMO, until a first transfer is effected, and the acronym ITMO thereafter.

The lifecycle of MOs and ITMOs is shown in Figure 3. The scope of these standards within this lifecycle is also depicted. The diagram is explained in the following paragraphs.

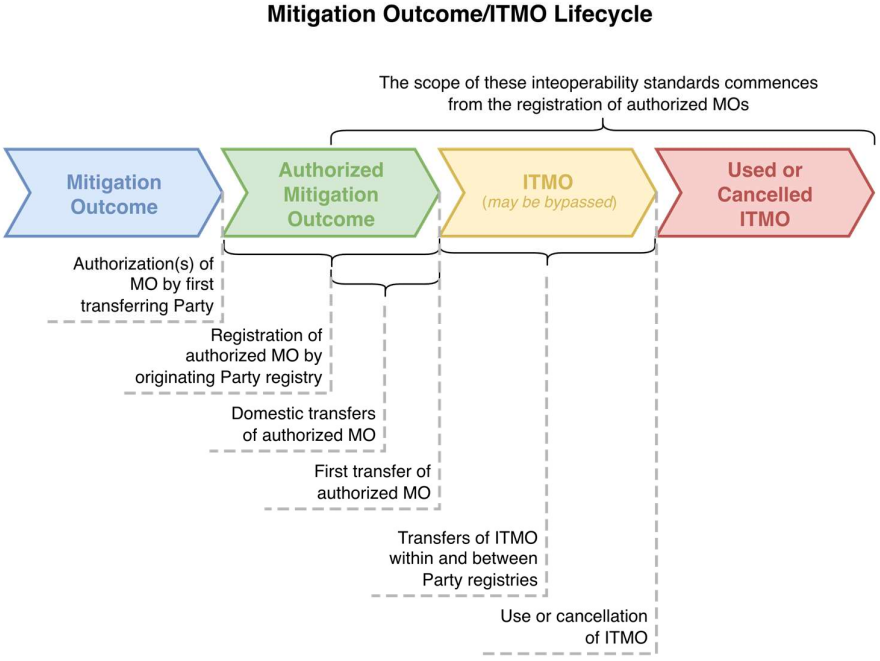

Figure 3 - Lifecycle of MOs and ITMOs

23. Mitigation Outcome Phase

(a) A mitigation activity (decision 2/CMA.3, annex, para. 10) generating emission reductions and removals (decision 2/CMA.3, annex, para. 1(b)), takes place in a host Party (decision 2/CMA.3, annex, para. 18(f)).

(b) The host Party is also the "first transferring Party" (decision 4/CMA.6, annex, table 3, footnote (e): first Transferring Party = "the participating Party in which the authorized mitigation outcome occurred as per common nomenclatures").

(c) The first transferring Party authorizes these mitigation outcomes (decision 4/CMA.6, annex, para. 11), and submits these authorizations to the secretariat, preferably using a voluntary standardized user-friendly template (decision 4/CMA.6, annex, para. 6).

(d) The secretariat processes the information contained in these authorizations and stores them on the CARP (decision 4/CMA.6, annex, para. 10). Authorizations are available to all registries (including those that do not opt to apply the communication standards) through API access to the CARP.

24. Authorized Mitigation Outcome Phase

(a) Each participating Party has, or has access to, a registry or a registry section in a consolidated system (decision 2/CMA.3, annex, para. 29). This choice should be documented by the participating Party in its initial report (decision 2/CMA.3, annex, paragraph 18).

(b) The authorized mitigation outcome is registered with interoperability hub by the originating Party registry.

(c) The authorized mitigation outcome may be transferred domestically.

(d) The first transfer of the authorized mitigation outcome occurs, by which the mitigation outcomes become ITMOs.

25. ITMO Phase

(a) ITMOs may be transferred within and between Party registries (within the limits set by the decisions) until they are used or cancelled. In some instances, the first transfer is already the last valid transaction in relation to an ITMO.

(b) ITMOs may also be used or cancelled by the holding registry, after which they become Used or Cancelled ITMOs. They may be:

- Used towards NDC (nationally determined contributions),
- Used towards IMP (international mitigation purposes),
- Cancelled for OP (other purposes),
- Cancelled for OMGE (overall mitigation of global emissions),
- Cancelled for administrative purposes, or
- Cancelled for a reason not described above.

(c) For authorized MOs with a first transfer definition of Use or Cancellation, this phase may be bypassed, when the Use or Cancellation first transfer moves the authorized MO directly to a Used or Cancelled ITMO.

26. Used or Cancelled ITMO Phase

(a) Used or Cancelled ITMOs may no longer be transferred.

B. Transactions

1. Transaction flows

27. A transaction may involve a single registry (e.g. use or cancellation), or two registries (e.g. transfer between two registries).

The flows are depicted in the sequence diagrams in the following subparagraphs. The diagrams provide a high-level view of the communication between the registries and the interoperability hub, and how the registries use the API operations provided by the interoperability hub. Importantly, they show there is no requirement for registries to implement any APIs (they will only be clients to interoperability hub's API) and the diagrams are not meant to dictate the internal processing of the registries.

(a) Single Registry Transaction Flow – Valid Proposal

The sequence diagram for valid proposals involving a single registry is depicted in Figure 4 (the sequence diagram for an invalid proposal is depicted in Figure 6).

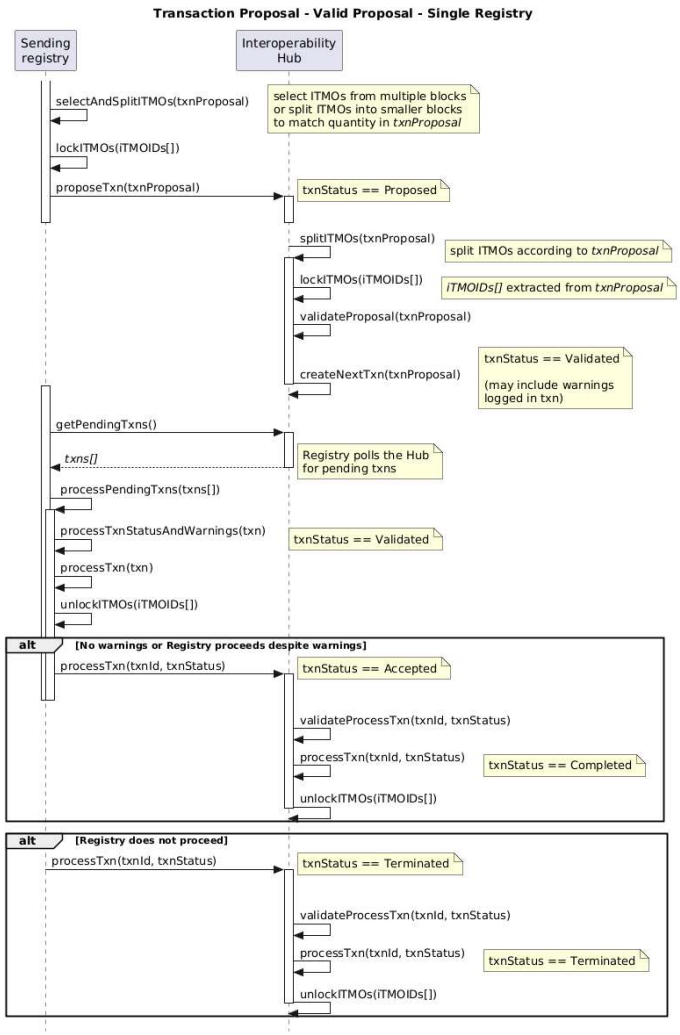

Figure 4 - Single Registry Valid Transaction Proposal Sequence Diagram

(b) Two Registry Transaction Flow – Valid Proposal

The sequence diagram for valid proposals involving two registries is depicted in Figure 5 (the sequence diagram for an invalid proposal is depicted in Figure 6)

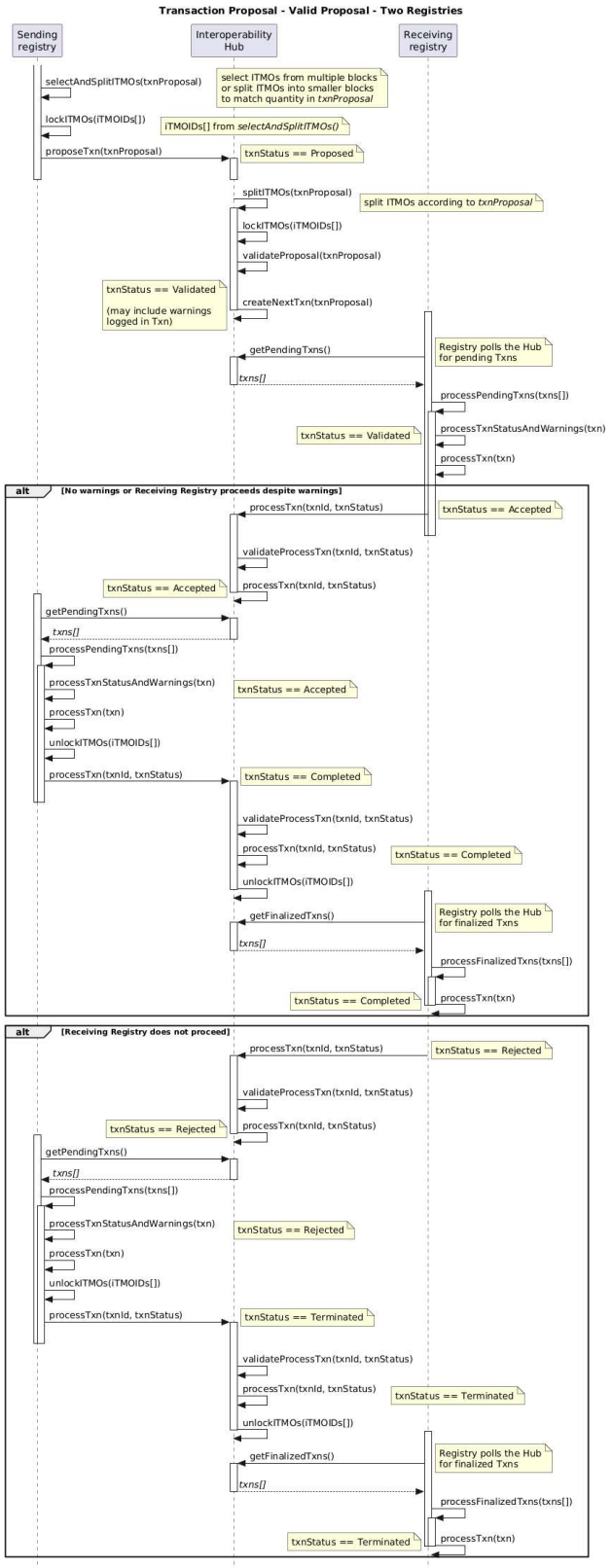

Figure 5 - Two Registry Valid Transaction Proposal Sequence Diagram

(c) Single and Two Registry Transaction Flow – Invalid Proposal

The sequence diagram for invalid proposals is depicted in Figure 6. This is the same flow regardless of *TransactionType*, as the proposal will be invalidated by the interoperability hub.

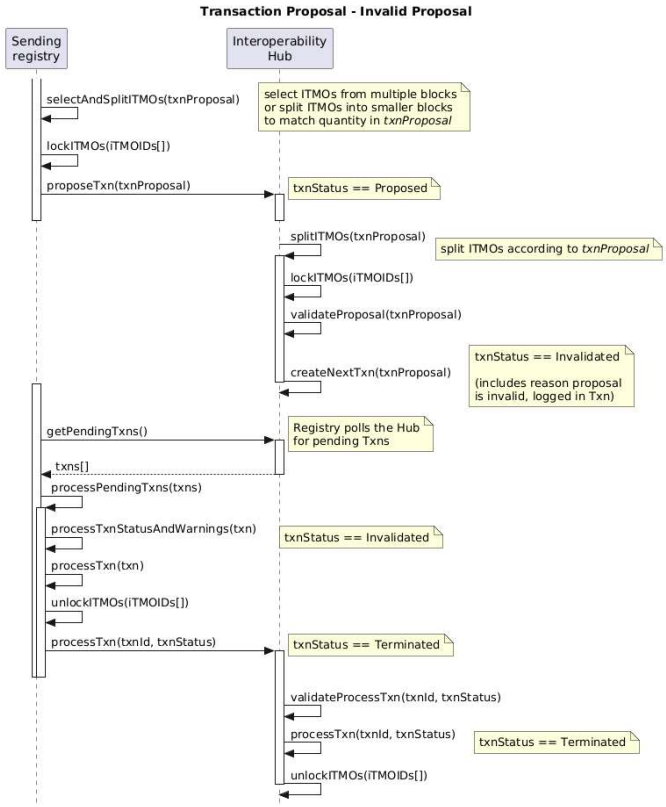

Figure 6 - Invalid Transaction Proposal Sequence Diagram

28. Transactions are initiated by registries using *proposeTxn()*.

29. Prior to calling *proposeTxn()*, a sending registry MUST:

(a) Select the ITMOs for the transaction (e.g. the appropriate quantity, vintage, cooperative approach);

(b) Split or merge blocks of ITMOs as needed;

(c) Lock the involved ITMOs so they are unavailable for other transactions;

(d) Verify, to the best of its ability, that the annex III validations will succeed.

30. On receipt of *proposeTxn()*, the interoperability hub MUST lock the ITMOs so they are unavailable for other transactions. If the ITMOs, or some subset of the ITMOs are already locked, the proposed transaction is deemed *Invalid* (*txnStatus* of the transaction is set to *Invalidated*).

31. The interoperability hub MUST validate a transaction proposed by a registry against the applicable annex III validations. If the transaction is validated, the interoperability hub MUST progress the transaction, possibly with warnings. If invalidated, the interoperability hub MUST return the reason(s) to the sending registry.

32. Registries MUST poll the interoperability hub to retrieve pending transactions in which they are involved, by using *getPendingTxns()*. It is recommended that this polling is periodic, with the following polling frequency:

(a) Every 10 minutes (standard interval), but

(b) At least once every 6 hours (to ensure updates are not missed), and

(c) No more than once every minute (to prevent overwhelming the interoperability hub with polling requests). The interoperability hub MAY be configured to ignore more frequent requests.

33. Once a registry has retrieved and processed a pending transaction, it MUST use *processTxn()* to notify the interoperability hub of the result of its processing: set the transaction status to *Accepted* or *Rejected*; to *Terminated* or *Completed*.

34. The receiving registry MUST *Accept* or *Reject* a validated transaction proposal and inform the interoperability hub of its decision which MUST then notify this decision to the sending registry. If there were warnings during the validation of the transaction, the interoperability hub MUST include them in the notification to the receiving registry.

35. When notified by the interoperability hub that a transaction it proposed has been *Accepted*, the sending registry MUST complete it. If notified that the transaction has been *Invalidated* or *Rejected*, the sending registry MUST *Terminate* it.

36. When the sending registry *Completes* a transaction, it MUST:

(a) Create or modify the relevant ITMOs as necessary;

(b) Unlock the involved ITMOs, so they are available for other transactions;

(c) Notify the interoperability hub of that the transaction is *Completed*.

37. When the sending registry *Terminates* a transaction, it MUST:

(a) Unlock the involved ITMOs, so they are available for other transactions;

(b) Notify the interoperability hub of that transaction is *Terminated*.

38. When the interoperability hub is notified by the sending registry that it has *Completed* a transaction, it MUST:

(a) Create or modify the relevant ITMOs as necessary;

(b) Unlock the involved ITMOs, so they are available for other transactions.

39. When the interoperability hub is notified by the sending registry that it has *Terminated* a transaction, it MUST unlock the involved ITMOs, so they are available for other transactions.

40. Registries MUST periodically (recommended every 10 minutes, but at least once every 6 hours) poll the interoperability hub to retrieve finalized transactions in which they are involved, by using *getFinalizedTxns()*.

41. Prior to retrieving the transaction with a *Completed* status, a receiving registry MUST NOT:

(a) Perform any processing on the transaction's ITMOs, and

(b) Propose further transactions on the ITMOs.

42. If the receiving registry retrieves a transaction with a *Terminated* status, and it had *Accepted* the same transaction, the receiving registry MUST *Terminate* the transaction.

43. If a transaction is not finalized within 24-hours of the latest status update by the interoperability hub, the interoperability hub MUST initiate a timeout process (see section 6 below).

2. Registration of authorised MOs

44. An originating Party registry MAY register authorised MOs in a Holding account.

45. Before authorised MOs are registered:

(a) The authorization of the MOs and their corresponding Cooperative Approaches must have been recorded with the CARP.

(b) The originating Party registry MUST provide a unique identifier for the authorised MOs to be registered. This consists of:

(i) The originating Party registry identifier,

(ii) First Id – the sequence number of the first MO in the block, and

(iii) Last Id – the sequence number of the last MO in the block.

The identifier must be unique – that is, the block represented by the first Id and the last Id must not overlap with any authorised MO block previously registered by the same originating Party registry.

When stored or communicated electronically, the unique identifier is comprised of the three elements above. However, when presented as a textual string, the three elements are separated by a hyphen "-" character, and the sequence numbers with thousands space " " separators. For example, "CHE01-1-3 142" represents the authorised MOs registered by the registry CHE01, with a first Id of 1, and a last Id of three thousand, one hundred and forty-two (3 142).

(c) The originating Party registry MUST not propose multiple registrations of the same, or overlapping authorised MOs before the first registration transaction has completed or is terminated.

46. Registrations are initiated by an originating Party registry via *proposeTxn()*, with a *TransactionType* of *Registration*.

47. The authorised MOs registered MUST be represented as blocks – each block identifying a serial number range. A registration request MUST specify one or more blocks.

48. Upon receiving a registration request from a registry, the interoperability hub SHALL, inter alia, validate that:

(a) All mandatory attributes of the authorised MOs are provided, as specified in annex I;

(b) The registry is not attempting the registration on behalf of another registry;

(c) The *fromAccount* and *toAccount* fields of the transaction are the same, and their *accountType* = 100 Holding.

(d) The authorised MOs being registered are unique – they do not overlap with another block of authorised MOs or ITMOs with the same originating Party registry;

(e) The authorised MOs do not overlap with an existing, ongoing registration.

49. The registration process MUST follow the single-registry transaction flow outlined in paragraph III.B.1.27(a) and SHALL be subject to the validations performed by the interoperability hub specified in annex III.

3. Transfer of ITMOs

50. A sending registry MAY propose a transaction to transfer ITMOs to a receiving registry, involving the movement of ITMOs represented as one or more blocks.

51. The proposed transaction MUST specify a sending account (to which the transferred ITMOs MUST be associated), and an existing receiving account, that MUST be different to the sending account.

52. The sending and receiving registry MAY be the same, in which case the transfer occurs between accounts of the same registry.

53. Upon receiving a transfer of ITMOs request from the sending registry, the interoperability hub SHALL, inter alia, validate that:

(a) The sending registry and account hold the ITMOs the registry wishes to transfer;

(b) The attributes of ITMOs to be transferred match the attributes known to the interoperability hub for the sending registry;

(c) The ITMOs to be transferred are not involved in another pending transaction (they are not locked by another transaction);

(d) The ITMOs to be transferred are not inconsistent because of an ongoing reconciliation process.

54. The interoperability hub SHALL also validate that the attributes of the ITMOs to be transferred match the most recent values communicated through the announcement(s) of modification of ITMOs attributes process. If the interoperability hub detects any difference, it SHALL:

(a) If the sending or the receiving registry has configured this validation is to be active via *configureValidation*(): Invalidate the transfer.

(b) If the sending and the receiving registry have configured this validation to be inactive via *configureValidation*(): append a warning to the transaction and affected ITMOs, informing the receiving registry of the difference(s) and send a textual message to the sending registry, informing it of the (potential) issue.

55. The interoperability hub SHALL forward any valid transfer request to the receiving registry, which MUST either accept or reject it.

56. The transfer of ITMOs process MUST follow the transaction flow outlined in paragraph III.B.1.27(b) and SHALL be subject to the validations performed by the interoperability hub specified in annex III.

4. Splitting and merging blocks

57. The interoperability hub MUST automatically split blocks associated with serial number ranges, to match the blocks proposed by a registry, as part of its processing of transactions.

58. To reduce fragmentation, a registry and the interoperability hub MAY, from time to time, merge blocks that share the same attributes and are contiguous. Merging of blocks inside a registry or inside the interoperability hub MUST NOT be communicated, as it is an internal representation of blocks within these systems.

5. Transaction statuses

59. A transaction MUST have one status at any given time, which is REQUIRED to be one of the following:

(a) *Proposed*: the transaction has been proposed by the sending registry and the ITMOs involved in the transaction are reserved exclusively for this transaction until it is finalized;

(b) *Validated*: the proposed transaction has been validated by the interoperability hub. As described in paragraph 68, some transaction validations are mandatory (will always be performed), others are configurable by each registry. Even if a registry configures a validation to be inactive, thus, the transaction status is *Validated*, despite the validation failing (e.g. the ITMO's authorization version is not the same as the most recent version in the CARP), this discrepancy is logged in the transaction, available for the registries to consider;

(c) *Invalidated*: the proposed transaction has been invalidated by the interoperability hub and MUST be terminated by the sending registry. An example is when the proposed ITMOs are currently locked by another concurrent transaction;

(d) *Accepted*: the receiving registry has accepted the valid transaction proposal. The receiving registry processes the ITMOs involved in the transaction, as necessary;

(e) *Rejected*: the receiving registry has rejected the valid transaction proposal;

(f) *Completed*: the previously *Accepted* or *Validated* transaction is finalized. The sending registry either removes the ITMOs or updates them as required and the updated ITMOs are available for further transactions;

(g) *Terminated*: the previously *Invalidated*, *Rejected* transaction is finalized. The sending registry does not update the ITMOs, but they are made available for further transactions. A transaction can also be *Terminated* if it is timed out by the interoperability hub (see section III.B.6), or reverted (see section III.B.7). A transition to the *Terminated* status MUST always be accompanied by a valid reason, which SHALL be one of the following:

(i) *Invalidated* by the interoperability hub;

(ii) *Rejected* by the receiving registry;

(iii) *Timed out* by the interoperability hub;

(iv) *Reverted*.

60. Registries and the interoperability hub MUST ensure that the topological order of the statuses of transaction, as shown in Figure 7 below, is always respected. In addition, registries and the interoperability hub SHALL ensure that timestamps for each transaction status change increase in order.

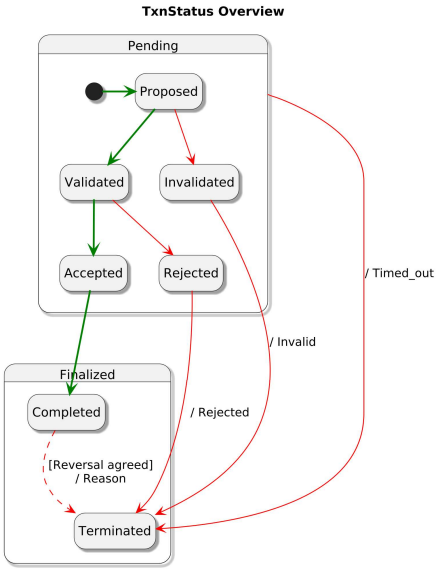

Figure 7 - Transaction status state diagram

61. The state diagrams in Figure 8 and Figure 9 show the relationships between the transaction statuses of the sending registry, receiving registry and the interoperability hub. Transitions related to transactions timeouts and reversal are omitted for simplicity.

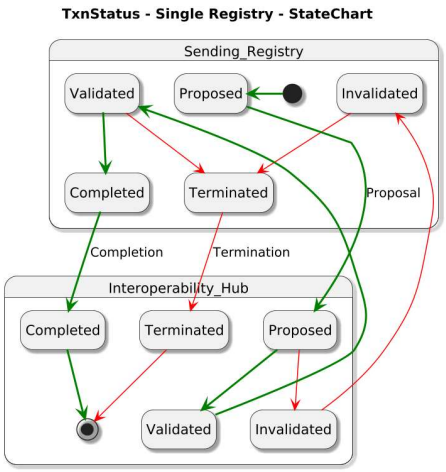

Figure 8 - Transaction state diagram - single registry transaction

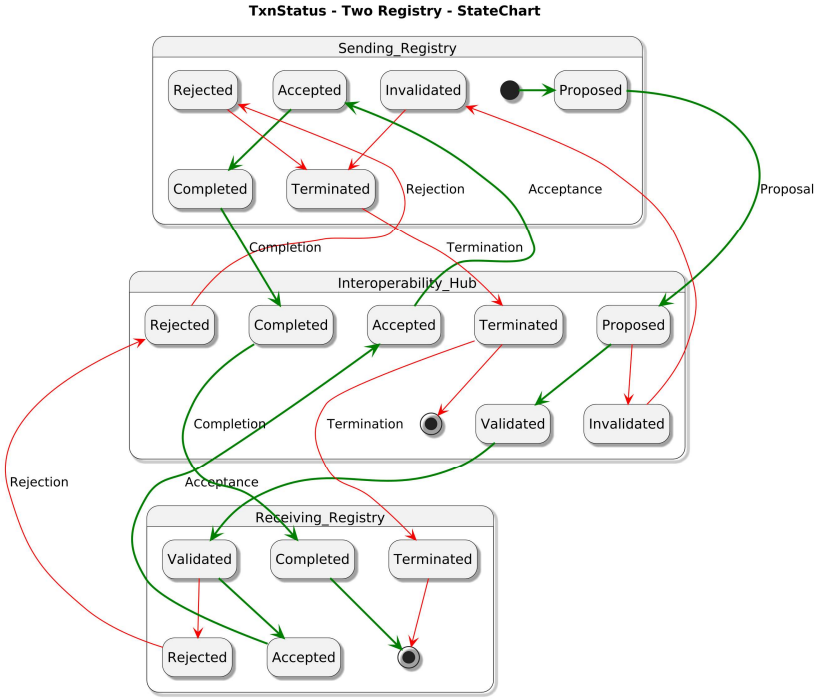

Figure 9 - Transaction state diagram - two registry transaction

6. Transaction timeout

62. If a transaction has not been completed within 24 hours after it was proposed, the interoperability hub MUST:

(a) *Terminate* the transaction with a reason of *Timed out*;

(b) Unlock the transaction's ITMOs to make them available for further transactions.

63. A registry's next successful *getFinalizedTxns()* call will retrieve transactions that have timed out, with a status of *Terminated*, the registry SHALL (see Figure 10 for sequence diagram):

(a) *Terminate* the transaction, indicating the reason as timed out and unlocking all ITMOs involved in the transaction so they become available for further transactions;

(b) Inform its registry administrator that the interoperability hub administrator MUST be contacted in all other cases (e.g. it already has been completed, which will lead to an inconsistency).

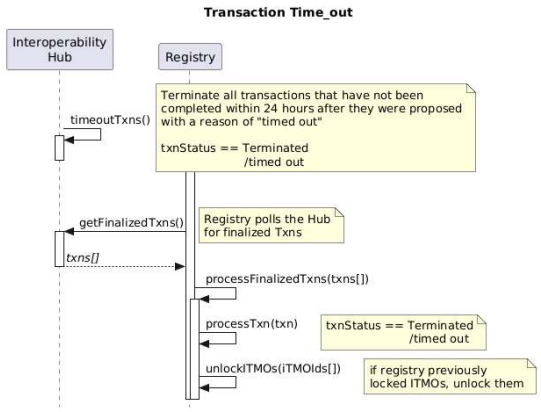

Figure 10 - Time-out Sequence

7. Reverting and correcting a transaction

64. In some situations, a registry administrator or the interoperability hub administrator MAY need to revert a previously completed transaction. These situations include accidental transaction proposals or resolving inconsistencies found during the reconciliation process. In this event:

(a) A manual review of the case MUST be conducted, coordinated by the interoperability hub administrator;

(b) If reverting the transaction is deemed feasible, the interoperability hub administrator MUST assign a unique identifier to the request: revertingId;

(c) Using the revertingId, the registry administrator and the interoperability hub administrator manually revert the transaction, setting it to *Terminated* status, with a reason of "Manually reverted by <revertingId> on <date/time>", and include in the reason any additional relevant details that justified the reversion.

(d) The registry MUST return any ITMOs involved to their previous state and initiate a reconciliation.

65. An operational procedure detailing the process of reverting and correcting transactions MUST be established. This procedure MUST detail the specific conditions under which reverting or correcting a completed transaction MAY occur.

8. Requesting transaction details

66. A registry MAY request the details of a single transaction from the interoperability hub using *getTxn*(*txnId*), subject to the visibility rules in para. 80 below, and the interoperability hub is REQUIRED to respond within 60 seconds with the transaction details, including its status. If a registry is only interested in the status of a transaction, it MAY use *getTxnStatus*(*txnId*).

67. A registry MAY request the details of multiple transactions that satisfy a filter criteria from the interoperability hub using *getTxns*(*mode*, *filter*), this is described in more detail in para. 85 below.

9. Transaction validation configuration

68. There are two types of transaction validations:

(a) Mandatory validations: these validations align with decisions and mandates, and cannot be turned off. If a transaction fails these validations, its *TxnStatus* MUST

be set to *Invalidated* by the interoperability hub. These validations are listed in Table 2 of annex III;

(b) Configurable validations: these are validations that SHOULD be used by all registries that help implement robust accounting within and between registries. While not intended to be a comprehensive set of business validations required by every registry, these validations reflect common practices and are included in the interoperability hub for ease of implementation. These validations, which can be configured, are listed in Table 3 of annex III. Each registry SHOULD also implement any additional business validations relevant to its Party and/or cooperative approach.

69. If a registry proposes a transaction that fails a configurable business validation, the interoperability hub MUST:

(a) Set the transaction *TxnStatus* to *Invalidated,* if the validation is enabled; or

(b) Keep the *TxnStatus* of the transaction unchanged and attach a warning about the failed validation to the transaction, if the validation is disabled.

70. *configureValidation*(*validationId*, *registryRole*, *isOn*) is used to configure a (configurable) validation:

(a) *validationId* specifies the validation to be configured (as listed in Table 3 of annex III),

(b) *registryRole* specifies when the configuration value is active: as the sending registry, as the receiving registry, or as either sending or receiving registry; and

(c) *isOn* specifies whether the configuration is to switch the validation on or off. Note that even if the validation is switched off, if a transaction fails the validation, the interoperability hub MUST attach a warning to the transaction.

C. Announcement of modification of ITMOs attributes

71. When any modification to attributes (e.g., authorization id, authorization version, vintage and/or mitigation outcomes) of ITMOs occur, the first transferring participating Party of the ITMOs SHOULD promptly announce the changes to the interoperability hub.

72. Upon receiving such announcement, the interoperability hub is REQUIRED to:

(a) Identify all affected ITMOs;

(b) Verify the changed values;

(c) If the change values verify, add a notification for all registries holding the affected ITMOs. These notifications SHALL indicate the affected ITMOs, the account identifiers holding the affected ITMOs, and the updated values of the attributes.

73. Upon retrieving the notification of modification of ITMO attributes, a registry SHOULD take appropriate transactions, which SHOULD include updating the relevant attributes of the affected ITMOs.

74. Once a registry acts on a notification by updating the ITMO, it SHALL acknowledge the notification by calling *acknowledgeNotification()*. Then, it will no longer receive notifications for that ITMO update from *getPendingNotifications()*, for example Holding Registry 1 in Figure 11.

75. On receiving the *acknowledgeNotification()* request from the registry, the interoperability hub SHALL update the relevant attributes of the ITMOs to maintain consistency with the holding registry.

76. If a registry does not act on a notification, for example, Holding Registry 2 in Figure 11, it will continue to receive notifications for that ITMO update from *getPendingNotifications()*. Further, when it attempts to transfer the ITMOs, the transaction

proposal will include a warning that attributes of the ITMO are inconsistent with that of the first transferring participating Party.

77. Once a registry no longer holds any affected ITMOs, notification for the ITMOs MUST no longer be created for it by the interoperability hub.

78. If a registry acquires ITMOs subject to a pending notification of modification, it SHALL receive broadcast announcements from the interoperability hub, noting that the ITMOs received will also contain a warning appended to them by the interoperability hub.

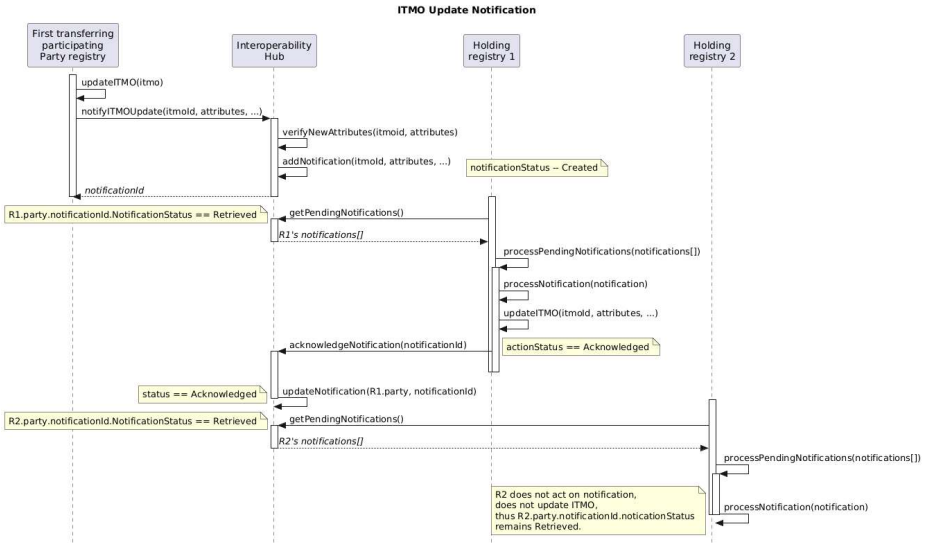

Figure 11 - Notification of update of ITMO attributes

D. Pulling and viewing holdings and transaction history of ITMOs

79. The interoperability hub MUST support the API operations listed in this section to allow for pulling and viewing of data and information on holdings and the transaction history of ITMOs.

80. When a calling registry retrieves information on transactions of ITMOs, the interoperability hub MUST enforce the following visibility rules:

(a) The calling registry MUST have proposed the transaction(s), or

(b) The calling registry MUST have received the transaction(s), if it was the receiving registry in a two-registry transaction flow and if the transaction(s) have reached the *Validated* status (i.e., *Invalidated* transactions are not visible to the receiving registry).

81. When a calling registry retrieves information on accounts, ITMOs and transactions of ITMOs, it MAY specify a mode as follows:

(a) Registry mode: returns the accounts created by-, the ITMOs held by-, and the transactions visible to-, the calling registry;

(b) Party mode (default): the accounts created by-, the ITMOs held by-, and the transactions visible to-, any of the registries belonging to the Party of the calling registry.

82. The interoperability hub MUST ensure that the retrieved information on accounts, ITMOs and transactions of ITMOs is browsable and filterable.

83. *getAccounts()* returns selected accounts, subject to the visibility rules above. The selected accounts MUST be returned in descending order based on their creation date (i.e., recently created accounts are returned first). Filter:

(a) Registry or Party mode: see above. If not specified, Party mode is assumed.

84. *getITMOs()* returns selected ITMOs, subject to the visibility rules above. The selected ITMOs MUST be returned in ascending order based on their identifier. Filters:

(a) Registry or Party mode: see above. If not specified, Party mode is assumed;

(b) Account identifier: an account identifier holding the ITMOs. If not specified, all ITMOs are selected.

85. *getTxns()* returns selected transactions, subject to the visibility rules above. The selected transactions MUST be returned in descending order based on their *Proposed* status date and time (i.e., recently proposed transactions are returned first). Filters:

(a) Registry or Party mode: see above. If not specified, Party mode is assumed;

(b) Transaction status: the transaction status (e.g. *Validated*) of the transactions to select. If not specified, all transaction statuses are selected;

(c) Minimum date and time: minimum date and time based on the *Proposed* status timestamp recorded by the interoperability hub. If not specified no minimum date and time applies;

(d) Maximum date and time: maximum date and time based on the *Proposed* status timestamp recorded by the interoperability hub. If not specified no maximum date and time applies;

(e) fromRegistry: only transactions from fromRegistry are returned. If not specified transactions from all registries are selected;

(f) fromAccount: only transactions from fromAccount are returned. If not specified transactions from all accounts are selected;

(g) toRegistry: only transactions to toRegistry are returned. If not specified transactions to all registries are selected;

(h) toAccount: only transactions to toAccount are returned. If not specified transactions to all accounts are selected.

E. Account management

86. A registry MUST declare to the interoperability hub the accounts it plans to use for transactions, either as sending or receiving accounts. Each account MUST have a type listed in tTable 5 of annex IV.

87. A registry SHALL NOT make any other information about the account(s) to the interoperability hub besides the account identifier(s) and account type(s), and the interoperability hub SHALL NOT store any other information about the account(s) besides the account identifier(s), account type(s) and its holdings. In particular, no information about the person(s) or entities owning the account SHALL be communicated or stored.

88. To manage its accounts, additionally to *getAccounts()* (see para. 83), a registry MAY call:

(a) *insertAccount()* to add a new account. If an account with the same account identifier already exists, the operation MUST fail;

(b) *removeAccount()* to remove an account. If the account holds ITMOs, or if the account is part of an ongoing transaction, the operation MUST fail.

F. Reconciliation management

1. Reconciliation between registries and the interoperability hub

89. Reconciliations between registries and the interoperability hub SHALL be based on the history of finalized transactions and on ITMO holdings as at 00:00 UTC of the current day. Pending transactions and/or transactions proposed and finalized after 00:00 UTC, and their resulting changes on ITMO holdings, MUST be ignored.

90. A registry SHOULD provide to the interoperability hub, daily, the list of its finalized transactions and ITMO holdings as at 00:00 UTC and request the interoperability hub to perform a comparison with its own transactions and holdings. Upon receiving this request, the interoperability hub MUST:

(a) Compare the list of transactions of ITMOs provided by the registry with its own, producing a list of differences (i.e., inconsistencies);

(b) Compare the ITMO holdings provided by the registry with its own, producing a list of differences (i.e., inconsistencies), while respecting the ITMO block boundaries of the registry;

(c) Return the list of differences of transactions and holdings to the registry, within 60 seconds of its request.

91. These standards provide *reconcile*(*txns*[], *itmos*[]) for this purpose:

(a) The registry's list of finalized transactions is provided in the *txns*[] list;

(b) The registry's holdings are provided in the *itmos*[] list;

(c) *reconcile()* returns a list of differences between the registry and the interoperability hub transactions, and a list of differences between the registry and the interoperability hub holdings.

92. The registry administrator of the registry and/or the interoperability hub administrator SHOULD promptly correct any inconsistencies in their respective systems, based on the comparison of the transactional history and inconsistent ITMOs, and any other pertinent elements.

2. Reconciliation within registries and within the interoperability hub

93. When a transaction is finalized, the interoperability hub MUST verify that its transactional history leads to the balance of ITMOS in each account (i.e., it performs an internal reconciliation of its records). Registries are RECOMMENDED to perform a similar verification. Registries MUST NOT communicate the results of these internal verifications to the interoperability hub.

3. Creating inconsistencies

94. In test environments, a registry MAY call *createInconsistency()* which will introduce the following inconsistencies in the interoperability hub:

(a) Create an additional *Completed* transaction and adjust the registry holdings accordingly;

(b) Delete an existing transaction without modifying the associated holdings;

(c) Add a block of ITMOs without generating the corresponding transaction;

(d) Modify the vintage of an ITMO without recording the corresponding transaction.

4. Operational procedure

95. An operational procedure detailing the process of reconciliation MUST be established.

G. Registry management

1. Interoperability hub status

96. *getServerStatus()* returns the operational status of the interoperability hub (i.e., operational, non-operational or transaction proposals are disallowed) along with its date and time.

2. Registry record creation within the interoperability hub

97. Once a registry administrator has configured its registry (or registry section) with a valid API key, *createRegistry()* MUST be used to create a new registry record in the interoperability hub, providing at least two valid contacts. In the production environment, the created registry record MUST belong to the Party linked to the API key used by the calling registry. In test environments, non-Party registries MAY be created. Upon creation, interoperability hub MUST set the registry operational status to non-operational.

3. Registry contacts

98. A registry SHOULD keep its contact information up-to-date by using *insertContact()* and/or *removeContact()*, in addition to specifying at least two contacts at the time of registry creation. The interoperability hub MUST use this information to keep the contact lists on the online platform accurate.

4. Registry operational status

99. A registry SHOULD notify the interoperability hub of its operational status using *setRegistryOpStatus(status)*. The operational status of a registry MUST be one of the following:

(a) Operational: all operations MUST be allowed;

(b) Non-operational: no operations MUST be allowed;

(c) Transactions disallowed: all operations MUST be allowed except for transaction processing.

100. The interoperability hub MUST invalidate a transaction involving a non-operational registry.

101. The interoperability hub MUST allow manual updates to the operational status of a registry.

102. Before it proposes a transfer of ITMOs, a registry SHOULD query the interoperability hub about the operational status of the receiving registry and SHOULD ensure that it is operational. *getRegistryOpStatus(regId)* returns the operational status of the registry identified by *regId*, according to the interoperability hub.

103. An operational procedure detailing the process of setting the operational status of a registry MUST be established.

5. Test registry reset and removal

104. *resetTestRegistry(regId)* MAY be used in test environments to reset a registry data. Calling this operation results in the removal of all holdings, transactions, accounts of the registry with identifier *regId*, including all ITMOs issued by the registry and held in other test registries, all transactions to/from the registry and all accounts it created. This operation MUST NOT alter the operational status of the registry, or its contacts. It SHALL NOT be available in the production environment.

105. *deleteTestRegistry(regId)* SHOULD be used to delete a test registry from the interoperability hub. Calling this operation results in the removal of all holdings, transactions, accounts of the registry with identifier *regId*, including all ITMOs issued by the registry and held in other test registries, all transactions to/from the registry and all accounts it created. It SHALL NOT be available in the production environment.

6. Registry onboarding

106. A registry SHALL be considered ready to undertake transactions in the production environment, when:

Manual initial setup:

(a) The Party of the registry has nominated at least two registry administrators;

(b) The registry administrator has access to the online platform;

(c) The registry IP address is known and whitelisted;

(d) The registry administrator has received the required API key(s) and configured them in its registry.

API operations initial setup:

(a) The communication between the registry and the interoperability hub is established and time of the registry and the interoperability hub is confirmed to be synchronized, using *getServerStatus()*. Confirmation that the time of the registry is synchronized is based on the interoperability hub administrator review of the request timestamp, and is provided as a textual message (see below);

(b) Textual message is confirmed working, using *getSelectedMsgs()*;

(c) The registry is created, using *createRegistry()*;

(d) The registry is in status operational, using *setRegistryOperationalStatus()*;

(e) Any additional registry contacts are available, using *insertContact()*;

(f) Transaction validations are configured, using *configureValidation();*

(g) The required accounts have been created, using *insertAccount()*.

H. Data export

1. Agreed Electronic Format

107. A registry MAY request a pre-filled AEF file export pursuant to decision 2/CMA.3, annex, chapter IV.B (Annual information). The exported file(s) are formatted in accordance with decision -/CMA.6, annex II.

108. The AEF template used during the export MUST be the one available at https://unfccc.int/documents/644483.

109. The cooperative approach and authorisation data required by the AEF reports are not fully stored in the interoperability hub. Instead, *exportAEFActions()* and *exportAEFHoldings()* retrieve the necessary information from the CARP.

110. *exportAEFActions()* exports AEF actions as an Excel file. The export includes only a filled-in "Table 3 Actions" AEF sheet. Only transactions with a *Completed* status and timestamp within this range MUST be included. The same visibility rules as in *getTxns()* (see para. D.85 above) MUST apply. Table 2 below maps transactions to AEF actions for action exports. Filters:

(a) Registry or Party mode: see para. D.81 above. If not specified, Party mode is assumed;

(b) Minimum date and time: minimum date and time of the transactions/actions to export, based on the *Completed* status timestamp recorded by the interoperability hub. If not specified, January 1st of the year preceding the current year applies (i.e., an AEF export of the previous calendar year is expected);

(c) Maximum date and time: maximum date and time of the transactions/actions to export, based on the *Completed* status timestamp recorded by the interoperability hub. If not specified, December 31st of the year preceding the current year applies (i.e., an AEF export of the previous calendar year is expected).

(d) Limit: the maximum number of actions in this export.

111. *exportAEFHoldings()* exports ITMO holdings as an Excel file. The export includes only a filled-in "Table 4 Holdings" AEF sheet. The calling registry MUST provide a cut-off date for the export, which represents the ITMO holdings on that date. The same visibility rules as in *getITMOs()* (see para. D.85 above) MUST apply. Filters:

(a) Registry or Party mode: see para. D.81 above. If not specified, Party mode is assumed;

(b) First transferred only: Only export ITMOs that have *isFirstTransferred* = TRUE; otherwise export all ITMOs.

(c) Cut-off date: date and time of the ITMOs holdings to export (i.e., the ITMO holdings is the result of all *Completed* transactions until the cut-off date). If not specified, December 31st of the year preceding the current year applies (i.e., an AEF export of the previous calendar year is expected).

Table 1 - Mapping transactions to AEF Actions

Transaction	**AEF Action**
Registration Same sending/receiving registries Same sending/receiving accounts	Not reported in AEF
Transfer Different sending/receiving registries	Acquisition Reported by receiving Party
Transfer Different sending/receiving registries	Transfer Reported by sending Party
Use Internal transfer to a "Use for NDC" or "Use for OP" account	Use The destination account type is used to infer the AEF Action subtype.
Cancellation Internal transfer to any "Cancellation" account	Cancellation The destination account type is used to infer the AEF Action subtype.
Not a transaction type, but indicated by the *isFirstTransfer* flag in the transaction, set by the registry that proposes the transaction.	First transfer

2. Preservation file

112. The interoperability hub SHALL export, on a monthly basis, all transactions, ITMO holdings, and any other pertinent information it holds in a long-term preservation file. One long-term preservation file SHALL be created per registry, in CSV format following RFC 4180, compressed to TAR and GZIP following POSIX.1-2001 and RFC 1952. A registry MAY request, from time to time, to receive its long-term preservation file. The long-term preservation file SHALL be sorted to allow for optimal compression.

113. An operational procedure detailing the process and practices related to data retention, including the precise content and format of the long-term preservation file, record-keeping arrangement and disaster recovery SHALL be established.

I. Textual message

114. A registry SHOULD be able to send textual messages to the interoperability hub using *sendMsg()*.

115. The interoperability hub MUST be able to send textual messages to one or more registry system administrators. Registries will retrieve these messages using *getSelectedMsgs()*.

116. A textual message MAY be accompanied by one or more attachment(s) in JPG, PNG or PDF format.

IV. Technical requirements

A. Time synchronization

117. Registries and the interoperability hub are REQUIRED to synchronize their time with UTC.

118. The timezone of timestamps described in these standards are UTC.

119. Registries SHALL provide the date and time at which each request originated in the HTTP header "Date", in accordance with section 5.6.7 of RFC 9110. The interoperability hub SHALL reject any request missing this header.

120. When a request is received, the interoperability hub MUST compare the time contained in the HTTP "Date" header with its own time. If the time difference is greater than 60 seconds, the interoperability hub MUST reject the request.

B. Timeouts and maximum payload size

121. Registries and the interoperability hub are REQUIRED to apply the following timeouts:

(a) (Registries) Connection timeout: 30 seconds;

(b) (Registries) Read timeout: 60 seconds;

(c) (Interoperability hub) Request read and processing timeout: 60 seconds.

122. The following HTTP maximum payload sizes SHALL apply:

(a) Request headers MUST NOT exceed 8190 bytes;

(b) Request body MUST NOT exceed 10 MB, unless otherwise specified;

(c) Response body MUST NOT exceed 10 MB, unless otherwise specified.

C. Security

123. Registry system administrators MUST be nominated by their respective national focal points. Once nominated, a registry system administrator MAY nominate additional contacts to support the operation of its registry. The interoperability hub administrator SHALL ensure that it only allows communications to and from registry system administrators and nominated contacts.

124. One-way TLS 1.3 MUST be implemented to guarantee the security of communications. The cipher suite TLS_AES_256_GCM_SHA384 (value: {0x13,0x02}, see RFC 8446, section 9.1 and Appendix B.4), MUST be used. The interoperability hub is REQUIRED to reject other versions of TLS and/or other cipher suites.

125. Each registry MUST have one static IP address, which MUST be whitelisted in the interoperability hub. Conversely, the interoperability hub MUST maintain one static IP address that is whitelisted in each registry.

126. When establishing a connection with the interoperability hub, a registry MUST validate public certificate of the interoperability hub by:

(a) Verifying its chain up to a trusted root certificate authority;

(b) Check its validity period;

(c) Verify its hostname;

(d) Check that its key usage and extended key usage are as expected;

(e) Validate its signature;

(f) Check it for revocation;

(g) Verifying that it has been generated using the agreed upon encryption algorithm(s).

127. API keys SHALL be used for authorization, in the form of tokens provided by registries in the HTTP request header "X-API-Key". HTTP Basic authentication SHALL NOT be used. The interoperability hub administrator is responsible for the management and distribution of API keys used in the registry system and SHALL operate an API key management tool as a sub-component of the interoperability hub.

128. API keys MUST follow the format: <Registry> - <Environment> - <Component > - <Nonce> - <Checksum>, where:

(a) Registry adheres to the format of annex I, section 1(a);

(b) Environment is "PROD" or "TEST";

(c) Component is "USER";

(d) Nonce is a cryptographically secure 256-bit random value, in hexadecimal, generated by the calling registry for each HTTP request;

(e) Checksum is used to verify the authenticity and integrity of the API key, using HMAC-SHA256, hexadecimal, truncated to the first 6 characters. The interoperability hub administrator SHALL manage, as part of the API key management tool, the shared secret keys used to create the checksum.

129. An example of a valid API key, using "SharedSecretKey" for checksum calculation, is:
CHE01-PROD-USER-ba24600bd49af0bf345213a9b6ec44e3-efe808 .

130. When the interoperability hub receives a request, it MUST reject it if:

(a) The registry IP address is not whitelisted;

(b) The API key provided is not valid for use, including:

(c) The IP address, API key, and identification of the registry in the request do not match.

131. If a registry provides a nonce has been used previously, or fails to provide a correct checksum three times, the interoperability hub SHALL automatically set that registry to non-operational status until the issue is investigated and resolved.

132. An operational procedure that is shared only with registry MUST be established, that will contain further details and procedures related to communication security and information security incidents.

D. Sandbox testing environment

133. The interoperability hub MUST provide a sandbox testing environment which registry administrators are RECOMMENDED to use for testing before deploying their registries to the production environment.

134. The sandbox testing environment SHALL allow registry administrators to:

(a) Create and remove, using *createRegistry()* and *deleteTestRegistry()*, other non-Party registries for testing transfer of ITMOs between them;

(b) Reset the content of both its test registry and of any non-Party registry it created, using *resetTestRegistry()*;

(c) Configure transaction validations, using *configureValidation()*;

(d) Create inconsistencies to test reconciliations, using *createInconcistency()*.

135. The interoperability hub MAY automatically clear the sandbox testing environment of a registry if there has been no activity for that registry for two weeks.

136. The security of the sandbox testing environment SHALL be adapted, as needed, to facilitate testing.

137. The usage of the sandbox testing environment SHALL be further detailed in an operational procedure.

E. Consolidated system

138. A registry MAY be hosted with other registries in a consolidated system (these registries are referred to as registry sections of the consolidation system) if it meets the following conditions:

(a) It MUST have a unique registry identifier;

(b) It MUST have its own static IP address and API keys;

(c) It MUST interoperate with other registries using these standards.

139. A registry administrator of a Party registry that is consolidated MAY delegate its authority to the administrator of the consolidated system it is part of.

F. Internationalization

140. Internationalization SHOULD be achieved as follows:

(a) The use of numerical identifiers and codes SHOULD be preferred to strings, where possible;

(b) When strings are used, they SHOULD use Unicode in accordance with JSON Schema Specification Draft 2020-12.

(c) Registry administrators are RECOMMENDED to provide localized versions of the numerical identifiers and codes within their registry.

G. Data retention and preservation

141. The interoperability hub is REQUIRED to retain all messages it receives from registries throughout its operational lifetime and for five years thereafter.

142. A registry SHOULD retain all messages it sends to the interoperability hub throughout its operational lifetime and for at least two years thereafter.

H. API standard and lifecycle management

143. These standards SHALL use the OpenAPI Specification 3.0.3.[2]

144. The endpoints APIs MUST be subject to lifecycle management, and SHALL adhere to the following principles:

(a) The OpenAPI “info.version” field is REQUIRED to be set to a string containing the major and minor version of the API (e.g. 1.0);

[2] https://swagger.io/specification

(b) The “basePath” field is REQUIRED to include the major version number and MUST be formatted as the letter “v” followed by major version number (e.g. /v1).

145. When a new API version has backwards-compatible changes, such as adding new fields or methods without modifying existing ones, the minor version MUST be incremented in the “info.version” field (e.g. 1.0, 1.1, 1.2 …).

146. When a new API version has changes to an existing method that breaks client code:

(a) The major version number in the “info.version” field MUST be incremented (e.g. 2.0, 3.0, …);

(b) The major version number in the “basePath” field. MUST be incremented (e.g., /v2, /v3, etc.).

147. These standards SHOULD aim to introduce changes at a low rate (preferably, none) and in a backward-compatible manner, as any disruptive change would severely impact registry systems. Consequently, the initial version of the APIs SHOULD be as comprehensive as possible.

148. The interoperability hub is REQUIRED to support multiple major versions concurrently, subject to coordinated deployments across registries. This process MUST be detailed in an operational procedure related to change management.

Annex I – Attributes of ITMOs

1. ITMO unique identifier (mandatory)

1. An ITMO identifier SHALL be composed of an Originating Party registry Id and a serial number range (*firstId*, *lastId*).

(a) Originating Party registry Id

2. An ITMO is REQUIRED to include the identifier the registry that registered it (as an authorised MO, in paragraph 23), referred to as the originating Party registry Id. This identifier is composed of two parts:

(a) The Party Id of the originating Party, based on the ISO 3166-1 alpha-3 country code, concatenated with

(b) A two-digit sequence number for the registry that is unique within each Party Id.

The Originating Party registry Id is stored as the concatenation of these two parts. For example, if the Party ABC had a single registry, the registry identifier would be "ABC01".

3. The originating Party registry Id is included in an ITMO's unique identifier because the originating Party registry is responsible for generating its unique identifier. This allows the originating Party registry to manage serial number ranges and ensure that all ITMOs it registers are distinct and do not overlap (i.e., are unique).

4. The following ISO 3166-1 alpha-3 user-assigned code elements SHALL be reserved:

(a) "XMR" for the Article 6.4 mechanism registry

(b) "XEU" for the European Union

5. For testing purpose, an ITMO MAY be issued by a non-Party registry. In such case, an ISO 3166 alpha-3 user-assigned code element MUST be used (e.g., AAB to AAZ, QMA to QZZ, XAA to XZZ, ZZA to ZZZ are available, with the exception of the user-assigned code elements reserved per para.4 above).

(b) First Id, Last Id – serial number range

6. An ITMO MUST have a *firstId* and a *lastId*, that represent the start and end of a block, inclusively (i.e., 1<= *startId* <= *lastId*).

7. For each Originating Party registry Id, serial number ranges MUST NOT overlap.

8. When stored or communicated electronically, the unique identifier is comprised of the three elements: originating Party registry Id, first Id, last Id. However, when presented as a textual string, the three elements are separated by a hyphen "-" character, and the sequence numbers use the space " " character as a thousands separator.

For example, "CHE01-1-3 142" represents the authorised MOs registered by the registry CHE01, with a first Id of 1, and a last Id of three thousand, one hundred and forty-two (3 142).

2. Cooperative approach Id (mandatory)

9. An ITMO is REQUIRED to have one cooperative approach. The *cooperativeApproachId* SHALL be a reference to the cooperative approach as published by the UNFCCC secretariat on the CARP.

3. First transferring Party (mandatory)

10. An ITMO is REQUIRED to have a first transferring Party. The *firstTransferringParty* of an ITMO SHALL be identified by its three-letter code as defined in ISO 3166-1 alpha-3 (e.g., "CHE" for Switzerland).

4. Vintage (mandatory)

11. An ITMO is REQUIRED to have a vintage year within the range 2021 to 2100, inclusive.

5. Unit type (mandatory)

12. An ITMO is REQUIRED to have a unit type equal to 1. This attribute is reserved for future use.

6. Authorization (mandatory)

(a) Authorization Id

13. An ITMO is REQUIRED to be authorized. The *authorizationId* SHALL be a reference to the ITMO Authorization as published by the UNFCCC secretariat on the CARP.

(b) Authorization version

14. An authorization MAY be updated after initial registration with the CARP. An ITMO is REQUIRED to be traceable to the version of the authorization that was registered. This will be through the ITMO's *authorizationVersion*.

7. Is First Transferred (mandatory)

15. Once an ITMO undergoes the first transfer, this flag is set, and cannot be unset, except if the proposed first transfer transaction of the ITMO is invalidated by the interoperability hub, or is rejected by the proposed acquiring Party. ITMOs that have been first transferred, cannot be subject to another first transfer.

8. Party ITMO registry Id (mandatory)

16. An ITMO MUST be trackable, thus, the current registry holding the ITMO is recorded in the ITMO. This is identified by the *partyITMORegistryId. Its format is the same as the one of point 3 (Originating Party registry Id) above*. Note that there are special cases where the registry holding the ITMO is not a Party registry (e.g., the Art 6.4 mechanism registry).

9. Account Id (mandatory)

17. The ITMOs SHALL be held in an account identified by *accountId*. Account Ids are unique within the registry they are associated with.

18. To minimize transcription errors, Account Ids SHOULD comply with the format of International Bank Account Numbers:

- Country code (using ISO 3166-1 alpha-2) of the Party holding the account – two letters,
- Basic Bank Account Number (BBAN) – 2 to 30 alphanumerical characters,
- Check digits, using MOD-97-10 as per ISO/IEC 7064:2003 – two digits.

10. Mitigation Type (mandatory)

19. An ITMO MUST have a mitigation type of "Emission reductions" or "Removals". Note that while the cooperative approach under which ITMOs are authorized may have a mitigation type of "Emission reductions and removals", each ITMO block MUST have a mitigation type of either "Emission reductions" or "Removals".

11. Underlying unit registry Id (optional)

20. An ITMO MAY have a unique identifier representing the underlying cooperative approach registry.

12. Underlying unit first Id (optional)

21. An ITMO MAY have a reference to the first unique identifier of the underlying unit block, which MUST be provided if the underlying unit registry Id is provided.

13. Underlying unit last Id (optional)

22. An ITMO MAY have a reference to the last unique identifier of the underlying unit block, which MUST be provided if the underlying unit registry Id is provided.

Annex II – Attributes of transactions

1. Transaction Id

1. A transaction SHALL be tracked by its Transaction Id. The Transaction Id is assigned by the interoperability hub when the transaction is proposed, and communicated back to the proposing registry.

2. From Registry

2. The registry identifier of the registry that currently holds the ITMOs.

3. From Account

3. The account identifier of the account, in the *fromRegistry*, that currently holds the ITMOs.

4. To Registry

4. The registry identifier of the registry that will hold the ITMOs after the successful completion of the transaction. Note the *toRegistry* may be the same as the *fromRegistry*.

5. To Account

5. The account identifier of the account, in the *toRegistry*, that will hold the ITMOs after the successful completion of the transaction. Note the *toAccount* is the same as the *fromAccount* in the case of *Registration* transactions.

6. Is First Transfer

6. The *isFirstTransfer* flag is set to TRUE if the transaction is a first transfer.

7. Txn Type

7. The type of transaction [*Registration*, *Transfer*, *Use*, *Cancellation*].

8. ITMOs

8. The array of ITMOs that are the subject of the transaction. This could be a single element array, representing a single block of ITMOs, or a multi-element array, representing multiple blocks of ITMOs.

9. Status

9. The status of the transaction, described in para. 59.

10. Status Date Time

10. The UTC date-time stamp of the last status change.

11. Errors

11. An array of validation errors that have caused the proposed transaction to be invalidated. These validations are described in annex III.

12. Warnings

12. An array of warnings that have been generated as a result of failing validation checks by the proposed transaction. The registry has configured these validations not to invalidate the proposed transaction, but to append warnings to the transaction instead. These validations are described in annex III.

13. Labels

13. Optional labels that may be attached to transactions.

14. External Reference

14. External reference information, such as a contract number.

15. Supp Info

15. Any other supplementary information.

16. Last Modified Date Time

16. The UTC date-time stamp of the last time any attribute of the transaction was modified.

17. Creation Date Time

17. The UTC date-time stamp of the time the transaction was created in the proposing registry.

Annex III – Transaction validations

I. Mandatory validations

1. The mandatory validations referred to in para. 68, which are carried out by the interoperability hub when it receives a transaction proposal, are detailed in Table 2 below.

Table 2 - Mandatory validations

Validation Id	*Transaction process*	*Validation description*
M001	All	The sending and receiving registries must exist.
M003	All	The sending and receiving registries must be operational.
M005	All	The sending and receiving accounts must exist.
M007	All	The ITMOs that are the subject of the transaction must not be involved in another transaction.
M008	All	Transactions proposed by a registry must have a strictly increasing creation date-time stamp.
MA001	Account management	A participating Party registry can only contain accounts with an account number whose IBAN country code is the Party of the registry.
MR001	Registration	A registration of authorised MOs must specify the same sending and acquiring registry.
MR002	Registration	A registration of authorised MOs must specify the same sending and acquiring account.
MR003	Registration	Authorised MOs must be registered to a 100 Holding account.
MR004	Registration	The authorised MOs registered must be unique (i.e. not previously registered).
MR005	Registration	A single registry must not propose multiple registrations concurrently.
MF001	First transfer	An ITMO that has already undergone first transfer must not be subject to another first transfer.
MF002	First transfer	For a first transfer transaction, both the *isFirstTransfer* flag of the proposed transaction, and the *isFirstTransferred* attribute of the authorised MO being first transferred, must be set in the proposed transaction.
MT001	Transfer	The ITMOs transferred must be held by the sending registry and account.
MT002	Transfer	A transfer or acquisition of ITMOs must specify a different sending and acquiring account.
MT003	Transfer	Transfers of ITMOs between registries must only occur between 100 Holding accounts.
MT004	Transfer	ITMOs that have been used or cancelled cannot be subject to further transactions.
MT005	Transfer	An ITMO must only be used or cancelled from a 100 Holding account located in the same registry.
MT006	Transfer	The attributes of the ITMOs transferred must match the interoperability hub attributes.

II. Configurable validations

2. The configurable validations referred to in para. 68, which are carried out by the interoperability hub when it receives a transaction proposal, are detailed in Table 3 below.

Table 3 - Configurable validations

Validation Id	*Transaction process*	*Validation description*
C001	All	An ITMO must refer to the most recent version of its authorization, as recorded in the CARP. *getAuthorizationVersions(authorizationId)* can be called on the CARP to retrieve the versions of an authorization, including the last version.
C002	All	Transaction MUST be consistent with the purpose(s) and first transfer definition(s) for which the MOs were authorized. *getAuthorizationPurposes(authorizationId, authorizationVersion)* can be called on the CARP to retrieve the purpose(s) and *getAuthorizationFTDefn(authorizationId, authorizationVersion)* can be called get the first transfer definition(s) contained in the authorization. The valid transactions, according to the authorization purpose(s) and definition(s) are shown in Table 4 below.
CR001	Registration	The authorized MO being registered must refer to an existing authorization identifier and authorization version, as recorded in the CARP. *getAuthorizationVersions(authorizationId)* can be called on the CARP to determine the existence of an authorisation and authorisation version. No other CARP based validations will be performed if this validation fails.
CR002	Registration	The serial numbers of registered authorized MOs for a first transferring Party must start with one and be contiguous.
CR003	Registration	The *originatingPartyRegistryId* attribute of the authorized MOs must match the Party of the sending registry (i.e., a Party registry is not allowed to register authorized MOs on behalf of another Party registry).
CR004	Registration	The Party within the *originatingPartyRegistryId* attribute of the authorized MOs must match the *firstTransferringParty*.
CR005	Registration	The quantity of authorized MOs registered must not exceed the quantity being authorized, as recorded in the CARP. *getAuthorizationQuantity(authorizationId, authorizationVersion)* can be called on the CARP to retrieve the quantity of MOs authorized.
CR006	Registration	Authorized MOs cannot be registered before the start date specified in their authorization, as recorded in the CARP. *getAuthorizationStartDate(authorizationId, authorizationVersion)* can be called on the CARP to retrieve the start date of the authorization.
CR007	Registration	Authorized MOs cannot be issued after the end date specified in their authorization, as recorded in the CARP. *getAuthorizationEndDate(authorizationId, authorizationVersion)* can be called on the CARP to retrieve the end date of the authorization.
CR008	Registration	The cooperative approach of the registered authorized MOs must be the one defined in the authorization, as recorded in the CARP. *getAuthorizationCAId(authorizationId, authorizationVersion) can* be called on the CARP to retrieve the cooperative approach identifier.
CR009	Registration	The vintage must be one of the vintages of the authorization, as recorded in the CARP. *getAuthorizationVintages(authorizationId, authorisationVersion)* can be called on the CARP to retrieve the vintages of an authorization.
CF002	First transfer	If the ITMOs in the proposed transaction have not been first transferred, and the proposed transaction matches with the purpose(s) and definition(s) for which the ITMOs were authorized, the transaction MUST be a first transfer – the first transfer flag of the proposed transaction MUST be set.
CT001	Transfer	The attributes of the ITMOs transferred must match the most recent announcement of modification by the first transferring Party.

III. Valid transactions

3. Table 4 lists the valid transactions for authorized MOs before their first transfer, and ITMOs after their first transfer, depending on the purpose(s) and first transfer definition(s) contained in their authorization.

Note that authorizations can have multiple purposes (e.g. NDC + IMP, NDC + OP, NDC + OIMP). When multiple purposes are authorized, the valid transactions shall include the aggregation of valid transactions for the combined purposes.

Table 4 is split into two parts:

- the top half describes the valid transactions for authorised MOs before they have been first transferred;
- the bottom half describes the valid transactions for ITMOs after they have been first transferred.

The columns on the left of the table specify the purpose and first transfer definition of the authorization, with one purpose and first transfer definition (if applicable) per row, marked with a "x". For each purpose and first transfer definition, the valid transactions are marked with a "✔", or a "①" if it is valid only for a first transfer (i.e. the transaction must be flagged as a first transfer)

	Authorization						Valid Transactions								
	Authorized Purpose			First Transfer Definition											
	NDC	IMP	OP	Authorization	Issuance	Use or Cancellation	Registration of authorized MOs from: (100) Holding account to: (100) Holding account of own registry	Domestic transfer from: (100) Holding account to: (100) Holding account in a registry of the same Party	International transfer[3,4] from: (100) Holding account to: (100) Holding account of another Party registry	Use towards NDC from: (100) Holding account to: (200) Use towards NDC account of own registry	Use towards IMP from: (100) Holding account to: (210) Use towards IMP account of own registry	Voluntary cancellation for OP from: (100) Holding account to: a (300) Cancellation for OP account of own registry	Voluntary cancellation for OMGE from: (100) Holding account to: (310) Voluntary cancellation for OMGE account of own registry	Voluntary cancellation from: (100) Holding account to: (320) Voluntary Cancellation account of own registry	Administrative cancellation from: (100) Holding account to: (330) Administrative cancellation account of own registry
Before first transfer	x						✔	✔	①				①	✔	✔
		x		x			①							✔	✔
		x			x		①							✔	✔
		x				x	✔	✔	①		①		①	✔	✔
			x	x			①							✔	✔
			x		x		①							✔	✔
			x			x	✔	✔	①			①	①	✔	✔

[3] Transfers to and from accounts in the mechanism registry (e.g. the adaption fund) with different IBAN country codes are considered international transfers for the purposes of these standards.

[4] Transfers to and from accounts in the mechanism registry with different IBAN country codes are considered international transfers for the purposes of these standards.

	Authorization						Valid Transactions								
	Authorized Purpose			First Transfer Definition											
	NDC	IMP	OP	Authorization	Issuance	Use or Cancellation	Registration of authorized MOs from: (100) Holding account to: (100) Holding account of own registry	Domestic transfer from: (100) Holding account to: (100) Holding account in a registry of the same Party	International transfer[3,4] from: (100) Holding account to: (100) Holding account of another Party registry	Use towards NDC from: (100) Holding account to: (200) Use towards NDC account of own registry	Use towards IMP from: (100) Holding account to: (210) Use towards IMP account of own registry	Voluntary cancellation for OP from: (100) Holding account to: a (300) Cancellation for OP account of own registry	Voluntary cancellation for OMGE from: (100) Holding account to: (310) Voluntary cancellation for OMGE account of own registry	Voluntary cancellation from: (100) Holding account to: (320) Voluntary Cancellation account of own registry	Administrative cancellation from: (100) Holding account to: (330) Administrative cancellation account of own registry
After first transfer	x							✔	✔	✔			✔	✔	✔
		x		x				✔	✔		✔		✔	✔	✔
		x			x			✔	✔		✔		✔	✔	✔
		x				x		✔	✔		✔		✔	✔	✔
			x	x				✔	✔			✔	✔	✔	✔
			x		x			✔	✔			✔	✔	✔	✔
			x			x		✔	✔			✔	✔	✔	✔

Table 4 - Valid transactions, before and after first transfer, depending on authorization

Annex IV – Account types

1. The types of accounts that registries may utilize to register and hold authorized MOs; and hold, transfer, use or cancel ITMOs are listed in Table 5 below.

Table 5 - Account types

Code	*Name*	*Description*
100	Holding	Authorised MOs are registered to 100 Holding accounts. ITMOs are transferred to and from 100 Holding accounts. *See decision 2/CMA.3, annex, para. 29.* Note: if a Party registry requires Pending and/or Authorized entity holding accounts, or additional types of holding accounts, it is its responsibility to determine how it will identify these account subtypes within its registry, as long as it transfers ITMOs to and from other registries using 100 Holding accounts. Also, Pending, Holding, Party holding, Authorized entity holding, and Share of proceeds of adaptation accounts in the Art 6.4 mechanism registry are 100 Holding accounts from the point of view of these standards (see decision 3/CMA.3, annex, paras. 58, 60 and 63).
200	Use towards NDC	When an ITMO is used towards NDC, it is transferred from a 100 Holding account to a 200 Use towards NDC account located in the same registry. *See decision 2/CMA.3, annex, para. 29.*
210	Use towards international mitigation purposes (IMP)	When an ITMO is used towards IMP, it is transferred from a 100 Holding account to a 210 Use towards international mitigation purposes (IMP) located in the same registry. *See decision 2/CMA.3, annex, para. 29 and decision 4/CMA.6, annex, table 3, footnote u.*
300	Cancellation for other purposes (OP)	When an ITMO is cancelled for other purposes (OP), it is transferred from a 100 Holding account to a 300 Cancellation for other purposes (OP) account located in the same registry. *See decision 2/CMA.3, annex, para. 29 and decision 4/CMA.6, annex, table 3, footnote u.*
310	Voluntary cancellation of ITMOs not counted towards NDC or OIMP, for OMGE	When ITMOs not counted towards NDC or OIMP are voluntarily cancelled for OMGE, it is transferred from a 100 Holding account to a 310 Voluntary cancellation of ITMOs not counted towards NDC or OIMP, for OMGE, located in the same registry. *See decision 2/CMA.3, annex, para. 39 and.*
320	Voluntary cancellation	When an ITMO is cancelled voluntarily for another reason than the ones specified in account type 300, 310 and 330, it is transferred from a 100 Holding account to a 320 Voluntary cancellation account, account located in the same registry.

Code	*Name*	*Description*
		See decision 2/CMA.3, annex, para. 29 and decision 3/CMA.3, annex, para. 61.
330	Administrative cancellation	When an ITMO is cancelled for administrative reasons, it is transferred from a 100 Holding account to a 330 Administrative cancellation account located in the same registry. *See decision 2/CMA.3, annex, para. 29.*

[a] The Art 6.4 mechanism registry will also use 340 Mandatory cancellation of A6.4ERs for OMGE (see decision 3/CMA.3, para. 59) and 350 Voluntary cancellation of A6.4ERs for OMGE (see decision 3/CMA.3, annex, para. 63 and 70).

Annex V – Definitions

Account: An account is a partition of a given registry which is used to hold related ITMOs.

API: An Application Programming Interface is a set of rules or protocols that enables software applications to communicate with each other to exchange data, features and functionality.

API operation: An API operation is a specific action that an API allows a client to perform. It typically corresponds to an HTTP request.

API key: An API key is a unique identifier used to authenticate and authorize access to an API

Atomicity: Atomicity is a property of database transactions that ensures that within a set of database operations, either all occur, or none occur.

Authentication: Authentication is the process to confirm the identity of a user.

Authorization: Authorization is the process to verify a permission to do something.

Block: A block is a sequential grouping of consecutive ITMOs.

Broadcast: A broadcast refers to sending a message, event, or data update to multiple recipients or systems at once.

Cipher suite: In TLS, a cipher suite is a set of cryptographic algorithms used to secure communications over a network. It comprises several components, each specifying different aspects of the encryption, authentication, and key exchange processes.

Client: A client is any computer hardware or software device that requests access to a service provided by a server

Concurrency: Concurrency is when two or more events or circumstances occur simultaneously.

Digital certificate: A digital certificate is a file or electronic password that proves the authenticity of a device, server, or user through the use of cryptography and the public key infrastructure (PKI).

Discrepancy: A discrepancy is an unexpected difference. For example, unexpected transactional data or timeouts could be considered discrepancies in a registry system.

First transferring Party: The first transferring Party of a mitigation outcome is the Party that authorizes the mitigation outcome[5].

HTTP Basic Authentication: HTTP Basic Authentication is an authentication scheme built into the HTTP protocol. It allows a client to provide a username and password when making a request to a server.

HTTP Client: An HTTP Client is a client that is able to send a request to and get a response from the server in HTTP format.

HTTP Server: An HTTP Server is a type of server responsible for serving or providing web content on the internet or intranet. As a part of a computer network, web servers transfer documents to clients.

GNU: "GNU is an operating system that is free software—that is, it respects users' freedom. The GNU operating system consists of GNU packages … as well as free software released by third parties." (from https://www.gnu.org)

GNU zip: GNU's version of the zip data compression tool.

Inconsistency: An inconsistency refers to a contradiction or discrepancy between different elements, data, or statements within a system, document, or context.

[5] Baku, FCC/PA/CMA/2024/L.15
para. 11: "*Clarifies* ... mitigation outcomes … if they have been authorized by the first transferring Party";
para. 12: "*Decides* …, where the first transferring Party has authorized the use of the same mitigation outcome …"

Interoperability hub: The interoperability hub is a component that tracks and verifies transactions of ITMOs to ensure consistency across connected registries.

ISO 3166-1 alpha-3: ISO 3166-1 alpha-3 are three-letter country codes defined in ISO 3166-1, part of the ISO 3166 standard published by the ISO, to represent countries, dependent territories, and special areas of geographical interest.

ISO 32000-2:2020: ISO 32000-2:2020 is an international standard which specifies what constitutes a well-defined PDF document, how PDF components are constructed and any expected behaviour from software consuming the PDF file.

ITMO: Internationally Transferred Mitigation Outcome. See decision 2/CMA.3, Annex, para. 1.

JSON: JavaScript Object Notation is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of name–value pairs and arrays (or other serializable values). It is a commonly used data format with uses in electronic data interchange, including that of web applications with servers.

JSON Schema Specification Draft 2020-12: The JSON Schema Specification Draft 2020-12 is a standardized format used to define the structure, constraints, and validation rules for JSON documents.

Key exchange: A key exchange is a method in cryptography by which cryptographic keys are exchanged between two parties, allowing use of a cryptographic algorithm.

Ledger: A ledger is a set of records used to track transactions, showing debits and credits for different accounts of an organization. It helps to ensure the accounts are consistent and balanced.

Lifecycle management: Life-cycle management, in the context of information technology, is the administration of a system from provisioning, through operations, to retirement.

Long-term preservation: Long-term preservation refers to the process of ensuring the accessibility, integrity, and usability of digital information over extended periods, typically involving strategies such as migration, emulation, and digital curation.

Manual intervention: A manual intervention is when an operational procedure is performed by hand rather than automatically enacted

Operational procedure: An operational procedure is a documented set of steps or guidelines that detail how a specific task or process should be carried out within the registry system. It provides a structured approach to executing activities, ensuring consistency, efficiency, and compliance with established standards or guidelines.

OpenAPI: The OpenAPI Specification is a framework used by developers to build applications that interact with REST APIs. The specification defines how to communicate with an API, what information can be requested, and what information can be returned.

Originating Party: The originating Party is the Party that owns the registry (the originating Party registry) that registers the authorized mitigation outcomes with the interoperability hub. In most cases, the originating Party is the first transferring Party.

Payload: In the context of data transmission and messaging protocols, a payload refers to the actual data being transmitted, excluding any protocol overhead or metadata.

Sandbox testing environment: A sandbox testing is a non-production environment which allows users to replicate baseline functionality without affecting the related mission critical system.

Private key: A private key is a cryptographic key that is used with an asymmetric (public key) cryptographic algorithm. The private key is uniquely associated with the owner and is not made public. The private key is used to compute a digital signature that may be verified using the corresponding public key.

Receiving account: A receiving account is the destination account to which ITMOs are sent.

Receiving registry: A receiving registry is the destination registry in each transaction.

Reconciliation: Reconciliation, when pertaining to data, refers to the verification checks surrounding the accuracy and consistency of said data.

Registry: A registry is an electronic system used to track, manage, and facilitate the trading of ITMOs. In the context of these interoperability standards, “registry" refers to national registries, including registry sections under the International Registry.

Registry administrator: The term registry administrator refers to the role performed by the individual or individuals nominated by the national focal point of the relevant Party to manage its registry. These individuals may include technical experts.

RFC 1952: RFC 1952 defines the structure and compression algorithm used in GZIP files.

RFC 2119: RFC 2119 defines key words that can be used to signify requirements in each specification and their interpretation.

RFC 3339: RFC 3339 defines a date and time format for use in Internet protocols.

RFC 4180: RFC 4180 defines the format and MIME type for CSV files, which are commonly used for representing tabular data in plain text form.

RFC 8446: RFC 8446 specifies version 1.3 of the Transport Layer Security protocol.

Sending account: A sending account is the origin account from which ITMOs are sent.

Sending registry: A sending registry is the origin registry for a given transaction.

Server: A server is a computer or system that provides resources, data, services, or programs to other computers, known as clients, over a network.

Transport Layer Security: Transport Layer Security is a security protocol designed to provide privacy and data security for communications over computer networks, such as the Internet. A primary use case of TLS is encrypting the communication between web applications and servers, such between web browsers and websites. The current version is TLS 1.3.

Token: A token is a small unit of data representing a particular piece of information or authorization granted to access a resource.

Trusted third party: A trusted third party is an entity which facilitates interactions between two parties who both trust the third party such as a certificate authority.

Unicode: Unicode is a text encoding standard maintained by the Unicode Consortium designed to support the use of text written in the world's major writing systems.

Unix: Unix is a family of multitasking, multiuser computer operating systems characterized by a hierarchical file system, shell-based command-line interface, and support for multitasking.

User-assigned code element: A user-assigned code element is a code at the disposal of users who need to add further names of countries, territories, or other geographical entities to their application of ISO 3166.

Vintage: A vintage is the calendar year in which the mitigation outcomes represented by the ITMO occurred.