Section 5b – VINAH transaction implementation guide

Victorian Integrated Non-Admitted Health (VINAH) minimum dataset manual

14th edition, July 2018

Version 1.0
Section 5b – VINAH transaction implementation guide

Victorian Integrated Non-Admitted Health (VINAH) minimum dataset manual

14th edition, July 2018

Version 1.0
Contents

Introduction .................................................................................................................. 1
How to use .................................................................................................................... 1
Conceptual entity integrity and keys ................................................................. 2
Transaction model ...................................................................................................... 3
Basic principles ........................................................................................................... 3
Message actions ........................................................................................................... 4
Insert ............................................................................................................................. 4
Updates .......................................................................................................................... 4
Deletes ......................................................................................................................... 4
Merges ........................................................................................................................... 4
Batches .......................................................................................................................... 5
Record rejection v Acceptance with batches ................................................... 5
Implementation approaches ..................................................................................... 6
Method 1 - Transaction checkpoint method ..................................................... 6
Exceptions .................................................................................................................. 6
Logical data structure .............................................................................................. 6
Logical structure data elements ............................................................................. 7
Checkpoint procedure ............................................................................................. 10
Implementation notes ............................................................................................... 14
Scope of selected data ............................................................................................. 14
Archiving, migrating or moving data ................................................................. 14
Ephemeral records .................................................................................................... 14
Non record-level errors ........................................................................................... 14
Submitted corrected rejections ............................................................................ 14
Resending previously accepted records ............................................................ 14
Record locking and concurrency issues ............................................................... 15
Patient merges ........................................................................................................... 15
Method 2 - Message/Transaction queuing .......................................................... 16
Method 3 - HL7 message interception ................................................................. 17
Data submission lifecycle ....................................................................................... 18
Overview ..................................................................................................................... 18
File transmission automation ............................................................................... 19
Uploading multiple files ........................................................................................ 19
Submission file naming .......................................................................................... 19
Submission file persistence .................................................................................. 20
Submission authorisation .................................................................................... 20
Character encoding ............................................................................................................. 20
File sizes ............................................................................................................................ 20

**VINAH submission reports** .......................................................................................... 21
Manual v automatic submission/acknowledgement ............................................................ 21
Consuming the submission report ...................................................................................... 21
Consumption process .......................................................................................................... 22
Waiting for a submission report ......................................................................................... 22
Persisting messages ............................................................................................................ 22
Non record level errors ...................................................................................................... 22
Errors that users cannot repair ............................................................................................ 23

**Purging submissions** .................................................................................................... 23

**Implementation considerations** ................................................................................... 24
Keeping track of transmissions ........................................................................................... 24
User data amendments ...................................................................................................... 24
Identifier schemes ............................................................................................................. 24
Program/Stream changes .................................................................................................... 24
Multiple source systems .................................................................................................... 25
Front end validations v VINAH validations ........................................................................ 25
Introduction

This section of the VINAH MDS manual specifies best practice for designing and implementing software to generate VINAH HL7 transmissions and maintain compliance with submission requirements over time.

How to use

Intended audience

This section of the manual is intended for use primarily by software designers and developers responsible for implementation of the VINAH MDS in the information systems used by agencies.

Use of this document

This document is a guide only. Compliance with all aspects of this implementation guide is not mandatory or measurable by the Department. Health care organisations may reference this standard when specifying requirements for their software.

Where this guide is inconsistent with other parts of the VINAH manual, those other parts will take precedence. If any errors or inconsistencies are found, please notify the HDSS Helpdesk.

Limitations

Applying principles outlined in this implementation guide either in part or in full may not guarantee compliance with the requirements of the data collection. This document is a guide for software developers to develop and understand the mechanics of the VINAH data submission lifecycle.

This document provides guidance around the best practices for VINAH submission process. The usage of methods outlined within this document does not imply acceptance of any product or process by the Department.

The Department will on request (where possible and appropriate) review specifications by vendors designing software systems and provide feedback but such review does not constitute acceptance, sign off or certification of those specifications.
Conceptual entity integrity and keys

All VINAH entities are related to one another through combinations of data elements that form foreign keys. These relationships are enforced in VINAH, and as such, each foreign key value must reference a record that has been sent and accepted into VINAH previously.

The Entity-Relationship diagram below shows the relationship between the conceptual entities and the data elements that will be used as their primary and foreign keys.
Transaction model

The VINAH transaction model is not simply an extract or ‘dump’ of activity that relates to a given period of time. The VINAH submission process is effectively an ongoing synchronisation of a set of data that exists at a health care organisation with a set of data at the Department.

In much the same way as a front end application writes inserts, updates and deletes to a database as a result of user input, VINAH manages data in a similar way.

The aims of the VINAH submission model include:
- only submitting additions or modifications to data, reducing the overall amount of data sent to the Department
- facilitate automated ‘background’ submission of smaller amounts of data more frequently
- receive information on data quality problems earlier in the data lifecycle, making correction easier.

Basic principles

The VINAH transaction model does not rely on developing an extract containing only information relating to a certain period of time. Rather, all relevant transactions in the health care organisation’s systems that occur between submissions are transmitted.

The following diagram outlines the difference between a period extract model and a transaction extract model as adopted in VINAH:

![Diagram showing the difference between Periodic Extract Model and Transaction Extract Model]
Message actions

Insert
Insert transactions insert data into the VINAH repository.
An insert will fail if a previous insert message has been sent and accepted.

Updates
Update transactions change existing data in the VINAH repository.
An update will fail if a previous insert message has not been sent and accepted.
An update cannot be used to perform an insert where the sending system is unsure whether an original insert has been sent and accepted.
In all cases, an update will overwrite data in all fields on the record it is updating. The implication of this is that all fields in the message must be populated with the correct data values regardless of whether they have changed or not; it is not acceptable to simply send the only field that has changed. The update is effectively a snapshot of the data as it exists in the sending system.

Deletes
Delete transactions remove existing data in the VINAH repository.
A delete message will fail if a previous insert message has not been sent and accepted.
In the case of deletes, all data content is ignored except the Key fields (e.g.: the Episode identifier on the Episode). However if data other than the key values are submitted they must be valid and meet business rules outlined in Section 4. Where records refer to parent or higher level records, the deletion of the higher-level entity will automatically delete all included lower-level entities; often termed a cascading delete. For example, when a Delete transaction is sent to delete an existing Episode, any contacts attached to the Episode will be removed from the VINAH repository. If it was desired to keep the contacts but attach them to a different episode, the submitting organisation system should, in the previous example, either send Contact update transactions to alter the Episode Identifier to refer to the new parent episode before sending the Episode delete, or send new insert contact messages with the new episode identifiers after sending the new Episode insert message.

Note: Where a submission that has performed deletes is rolled back (purged), the records deleted as a result of that submission will be reinstated; this includes any lower-level entities that were deleted automatically.

Merges
Merge messages will merge two previously accepted records into one. All records that refer to the Prior Identifier will be changed to point to the New Identifier. The Prior Record is then deleted, although it can be re-created through an insert messages after this point. If one or both of the Clients to be merged have not previous been submitted and accepted, then A04 messages for either or both of these clients will need to be included in the same submission file as the Merge message (A40).
The merge transaction is currently only applicable to the Patient/Client; merging other entities such as Episodes is not supported at this point in time.
The merge transaction will be rolled back and the records un-merged in VINAH when a file is purged. This should be taken into account when implementing submission rollback processes.
Batches

Batches within VINAH are Transaction Boundaries defined by the sending system that form areas in which groups of transactions succeed or fail together.

As a result, any errors that occur to any message inside the batch will result in all messages within the batch not being committed to the VINAH data store. This includes messages that preceded the failed message in that batch. It is important to note that messages in a batch may not necessarily be invalid or receive a rejection message, but their data will not be stored.

Using batches to group data in this manner is an effective way of isolating errors from other data in the file.

Batches can be created according to any desired methodology that suits the sending system. The suggested practice is for a batch per patient to be sent. It is not recommended to put all data into one batch per file, as this could likely result in the same valid data being excessively re-processed and revalidated during the error correction process, and prevent clean data from becoming available for VINAH reporting until invalid data is corrected.

Although not a strict requirement for VINAH, it is a best practice method to ensure that errors that may be affecting only one patient are isolated from other records in the file. As outlined in Section 5, a single batch is accepted/not accepted as a whole, so grouping all patient activity into one batch strikes the right balance.

Record rejection v Acceptance with batches

The primary effect of transaction boundaries is the distinction between explicit rejection and explicit acceptance of a record.

A record may not be accepted, but not rejected. A rejection is a result of a records violation of one or more business rules. All rejected messages are not accepted. However, if a record is in the same batch/transaction boundary as a rejected record, it may meet all the business rules but will not be accepted as the containing batch had failed.
Implementation approaches

The following section outlines some common implementation methods for implementing a VINAH submission lifecycle.

Method 1 - Transaction checkpoint method

The VINAH transaction checkpoint method maintains a long-term record of the data interchange between the data provider and the Department. It aims to allow the submitting system to know exactly which database action (Insert, Update or Delete) needs to be sent to the Department for any given record.

The VINAH transaction log is a concept that can be implemented in many ways. This guide will summarise a generic approach.

Exceptions

Although VINAH records are accepted or rejected within the scope of their containing batch (transaction boundary), this method does not explicitly deal with batches of records. The data structures outlined here may be extended to include information regarding batches sent, at the discretion of the implementer.

Full end-to-end VINAH implementation scenarios are demonstrated in the VINAH Demonstration Application available on request from the Department.

Logical data structure
Logical structure data elements

**internal_record_pointer**

A structure which provides references to various data within the local system. Using a pointer structure decouples the VINAH submission process from the local system. The record pointer structure need not be physically implemented; data structures in the local system can be directly referenced where:

- they carry the necessary metadata elements as outlined below, and
- they carry a stable unique key that is not recycled.

Where the record pointer is physically implemented, it may be populated in real time from the local data, or assembled at the point of VINAH transaction.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vinah_record_key</td>
<td>A Unique identifier for the VINAH Record over time. Where the internal_record_key is stable and unique (i.e. a Unit Record Number or equivalent) this can be used, or optionally a surrogate identifier can be generated. Identifiers can also be formed from other local data elements, which when concatenated, are made unique.</td>
</tr>
<tr>
<td>vinah_entity</td>
<td>A Department defined data entity, as defined in Section 2 (and refined in Section 3) e.g., Patient/Client, Episode etc., that maps to the internal_record_type.</td>
</tr>
<tr>
<td>internal_record_key</td>
<td>A locally generated key which identifies a record in the context of an internal_record_type.</td>
</tr>
<tr>
<td>internal_record_type</td>
<td>A locally-defined identifier of the type of record identified by the internal_record_key, e.g. the table name of the record or similar.</td>
</tr>
<tr>
<td>create_datetime</td>
<td>The date the record was created in the local system.</td>
</tr>
<tr>
<td>last_update_datetime</td>
<td>The date (if any) the record was last altered in the local system (may be the same as the create_date). Dates of individual updates may be collected but the date of the most recent update prior to the checkpoint is referenced in the VINAH transaction method.</td>
</tr>
<tr>
<td>delete_datetime</td>
<td>The date (if any) the record was deleted or became out of scope of VINAH reporting.</td>
</tr>
<tr>
<td>merge_datetime</td>
<td>The date (if any) the record and its related child-level data was merged with another record at the same level. For VINAH the only current merge implemented is at the patient level.</td>
</tr>
</tbody>
</table>
**vinah_transaction**

Reflects a log of the assembly of all relevant transactions to submit to VINAH.

<table>
<thead>
<tr>
<th><strong>transaction_id</strong></th>
<th>A suitable unique identifier of the instance of the VINAH transaction.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>checkpoint_datetime</strong></td>
<td>The last date that transactions were assembled for a VINAH submission. This may not reflect the actual date of submission of the file – instead it aims to record a moment in time at which all candidate transactions were assembled.</td>
</tr>
<tr>
<td><strong>filename</strong></td>
<td>The name of the file that was generated for the transaction checkpoint. Due to the file size limits in VINAH, this should be implemented in a way that supports the generation of multiple files for a given checkpoint, which may mean a separate physical structure to keep track of individual files generated by a transaction checkpoint.</td>
</tr>
<tr>
<td><strong>submit_datetime</strong></td>
<td>Optional; The date/time that the user submitted the packaged VINAH data. If the local system is not performing the submission itself, this information can be gleaned from the user, or from the submission report returned from VINAH.</td>
</tr>
<tr>
<td><strong>ack_datetime</strong></td>
<td>The date/time that VINAH completed the validation process and returned the submission report – this element can be mapped to the process_end_date element in the submission report (see Section 7). Note this element does not refer to the file acknowledgement receipt, which only indicates the file has been received by VINAH.</td>
</tr>
<tr>
<td><strong>&lt;other metadata&gt;</strong></td>
<td>Other metadata may be stored at the discretion of the implementer, in particular file names etc. It is suggested that security auditing information be stored to identify the user who performed the VINAH transaction.</td>
</tr>
</tbody>
</table>
**vinah_record_transaction**

Reflects a log of records that were assembled within a transaction.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record_x_id</td>
<td>An identifier of an instance of a record being sent to VINAH within a transaction.</td>
</tr>
<tr>
<td>transaction_id</td>
<td>A pointer to the vinah_transaction instance.</td>
</tr>
<tr>
<td>vinah_record_key</td>
<td>A pointer to the internal_record_pointer.</td>
</tr>
<tr>
<td>vinah_entity</td>
<td>A Department defined data entity, as defined in Section 2 (and refined in Section 3).</td>
</tr>
<tr>
<td>transaction_type</td>
<td>Indicating if the transaction type was an insert, update or delete.</td>
</tr>
<tr>
<td>record_transaction_datetime</td>
<td>The date that the record was inserted, updated or deleted in the local system. Used to order transactions correctly.</td>
</tr>
<tr>
<td>mcid</td>
<td>The Message control identifier of the instance of the record. The record_x_id field can be used as the Message Control Identifier, or alternatively it can be constructed in any fashion that ensures their uniqueness; i.e. concatenate several record transaction fields: left([transaction_type],1) &amp; '-' &amp; left([vinah_entity],3) &amp; '-' &amp; [vinah_record_key] &amp; '-' &amp; [record_x_id] with the resultant value i.e.: 'i-pat-10234-123123' The above example does not guarantee more uniqueness than the [record_x_id] field alone, but is useful when tracing messages sent to the Department as other relevant information is embedded in the identifier.</td>
</tr>
<tr>
<td>accepted_flag</td>
<td>Boolean flag set from the VINAH submission report indicating if the message was accepted or not. Note the difference between message acceptance and message rejection.</td>
</tr>
<tr>
<td>rejected_msg</td>
<td>A validation message as returned by the VINAH system. A null value would indicate there is no rejection on the message.</td>
</tr>
</tbody>
</table>
Checkpoint procedure

The following section contains some code examples in Transact-SQL (T-SQL) format. Each query outlined in this section may need to be repeated for each VINAH entity (e.g. referral in, patient, episode etc.).

1. Create transaction checkpoint
   Insert a record into the vinah_transaction table > populate the checkpoint_datetime with date/time the procedure was executed (or other point in time as necessary). Populate security metadata for audit purposes.

   INSERT INTO [vinah_transaction] ([checkpoint_datetime])
   VALUES (getdate())

2. Select candidate records affected since last checkpoint and current checkpoint
   The following pseudo SQL is applied at this level, with the select statements to return data listed within:

   INSERT INTO [vinah_record_transaction]
   ([transaction_id], [vinah_record_key], [vinah_entity], [transaction_type],
   [record_transaction_datetime], [mcid])

   2.1 Select deleted records - select records that have been deleted since the last checkpoint, where the insert transactions for those records have been previously sent to and accepted by VINAH

   SELECT @this_transaction_id, [vinah_record_key], [vinah_entity],
   ‘delete’, [delete_datetime]
   FROM [internal_record_pointer] del_records
   WHERE [delete_datetime] BETWEEN @last_checkpoint_datetime and @this_checkpoint_datetime
   AND EXISTS (SELECT [record_x_id]
               FROM [vinah_record_transaction]
               WHERE [vinah_record_key] = del_records.[vinah_record_key]
               AND [vinah_entity] = del_records.[vinah_entity]
               AND [transaction_type] = ‘insert’
               AND [accepted_flag] = 1
           )

   2.2 Select inserted records - select records that have been created since the last checkpoint, and were not deleted in the same period, where the insert transactions for those records have not been previously sent to and accepted by VINAH

   UNION SELECT @this_transaction_id, [vinah_record_key], [vinah_entity],
   ‘insert’, [create_datetime]
   FROM [internal_record_pointer] ins_records
   WHERE [delete_datetime] is null
   AND [insert_datetime] BETWEEN @last_checkpoint_datetime and @this_checkpoint_datetime
   AND NOT EXISTS (SELECT [record_x_id]
                   FROM [vinah_record_transaction]
                   WHERE [vinah_record_key] = ins_records.[vinah_record_key]
                   AND [vinah_entity] = ins_records.[vinah_entity]
                   AND [transaction_type] = ‘insert’
                   AND [accepted_flag] = 1
               )
2.3 Select updated records - select records that have been modified since the last checkpoint, where the insert transactions for those records have been previously sent to and accepted by VINAH

```
UNION SELECT @this_transaction_id, [vinah_record_key], [vinah_entity], 'update', [last_update_datetime]
FROM [internal_record_pointer] upd_records
WHERE [delete_datetime] is null
AND [update_datetime] (BETWEEN @last_checkpoint_datetime and @this_checkpoint_datetime)
AND EXISTS(
    SELECT [record_x_id]
    FROM [vinah_record_transaction]
    WHERE [vinah_record_key] = upd_records.[vinah_record_key]
    AND [vinah_entity] = upd_records.[vinah_entity]
    AND [transaction_type] = 'insert'
    AND [accepted_flag] = 1
)
```

2.4 Select merged records - Select Records that have been merged since the last checkpoint. As the merge operation has already been committed at the client end it cannot be simulated at a later date like the other transaction types. Hence this select statement does not check if either of the records involved in the merge have been previously accepted by VINAH, as the merge operation needs to happen at this point in time.

```
UNION SELECT @this_transaction_id, [vinah_record_key], [vinah_entity], 'merge', [merge_datetime]
FROM [internal_record_pointer] mer_records
WHERE [merge_datetime] (BETWEEN @last_checkpoint_datetime and @this_checkpoint_datetime)
```

3. Select candidate records that were not previously accepted

3.1 Select deleted records - select records that were sent as deletes in the last transaction and were a part of a batch of messages not accepted by VINAH (due to one or more messages within the containing batch being rejected). The records will be resent in this case even though they have not changed in the local system.

```
UNION SELECT @this_transaction_id, [vinah_record_key], [vinah_entity], 'delete', [delete_datetime]
FROM [internal_record_pointer] del_records
WHERE [delete_datetime] <= @last_checkpoint_datetime
AND EXISTS(
    SELECT [record_x_id]
    FROM [vinah_record_transaction]
    WHERE [transaction_id] = @last_transaction_id
    AND [vinah_record_key] = del_records.[vinah_record_key]
    AND [vinah_entity] = del_records.[vinah_entity]
    AND [transaction_type] = 'delete'
    AND [accepted_flag] = 0
)
```

3.2 Select inserted records - select records that were sent as inserts in the last transaction and were a part of a batch of messages not accepted by VINAH (due to one or more messages within the containing batch being rejected). The records will be resent in this case even though they have not changed in the local system.

```
UNION SELECT @this_transaction_id, [vinah_record_key], [vinah_entity],
```
'insert', [insert_datetime]
FROM [internal_record_pointer] ins_records
WHERE [insert_datetime] <= @last_checkpoint_datetime
AND [delete_datetime] is null
AND EXISTS (  
    SELECT [record_x_id]
    FROM [vinah_record_transaction]
    WHERE [transaction_id] = @last_transaction_id
    AND [vinah_record_key] = ins_records.[vinah_record_key]
    AND [vinah_entity] = ins_records.[vinah_entity]
    AND [transaction_type] = 'insert'
    AND [accepted_flag] = 0
)

3.3 Select update records - select records that were sent as modified in the last transaction and were a part of a batch of messages not accepted by VINAH (due to one or more messages within the containing batch being rejected). The records will be resent in this case even though they have not changed in the local system.

UNION SELECT @this_transaction_id, [vinah_record_key], [vinah_entity],
'update', [last_update_datetime]
FROM [internal_record_pointer] upd_records
WHERE [last_update_datetime] <= @last_checkpoint_datetime
AND EXISTS (  
    SELECT [record_x_id]
    FROM [vinah_record_transaction]
    WHERE [transaction_id] = @last_transaction_id
    AND [vinah_record_key] = upd_records.[vinah_record_key]
    AND [vinah_entity] = upd_records.[vinah_entity]
    AND [transaction_type] = 'update'
    AND [accepted_flag] = 0
  )

3.4 Select merge records – select merge records sent in the last transaction and were part of a batch of messages not accepted by VINAH (due to one or more messages within the containing batch being rejected). As merge message generally cannot be changed in the system, it is generally resent without change.

UNION SELECT @this_transaction_id, [vinah_record_key], [vinah_entity],
'merge', [merge_datetime]
FROM [internal_record_pointer] mer_records
WHERE [merge_datetime] <= @last_checkpoint_datetime
AND EXISTS (  
    SELECT [record_x_id]
    FROM [vinah_record_transaction]
    WHERE [transaction_id] = @last_transaction_id
    AND [vinah_record_key] = mer_records.[vinah_record_key]
    AND [vinah_entity] = mer_records.[vinah_entity]
    AND [transaction_type] = 'merge'
    AND [accepted_flag] = 0
  )
4. **Sort records**

The dataset resultant from the previous SQL SELECT statements can be sorted as such:

```sql
ORDER BY [record_transaction_datetime] ASC
```

This ensures that transaction records are created in the order in which they occurred.

5. **Apply batching**

Further sorting and grouping can be undertaken to assemble records together in the desired fashion. For a patient-per-batch approach, the result set could be grouped by patient identifier, then by ordered `record_transaction_datetime`. It is recommended that merge operations be isolated in their own batch to ensure the correct processing order.

6. **Write submission file(s)**

Generated messages for each batch in the appropriate format (e.g. HL7/XML), according to the rules of that format.
Implementation notes

Scope of selected data

The transaction checkpoint method does not explicitly deal with the business criteria for selecting data. The selection of records should be in line with the concepts defined in Section 2 of the manual. The transaction checkpoint method also allows the scope of data to be increased as requirements change. This extends to reporting historical data where new criteria are introduced; the record pointer structure can simply reference a wider range of records.

For data that has changed to no longer be in scope (for example data was incorrectly assigned to a program/stream, has been corrected and is subsequently not in scope) it is conceivable that the record pointers may no longer resolve to these records, even though VINAH record transaction entries still exist. When implementing this method an approach to handle the removal of data from scope (i.e. issuing delete messages if record pointers disappear, as opposed to if there is a delete date) may be considered.

Archiving, migrating or moving data

During the lifecycle of clinical data systems it is possible that historical data that is in the scope of the VINAH collection may be archived or moved to another system. Care should be taken to ensure any remaining data referring to the removed data maintains its referential integrity; for example ensure that if a patient is archived, episodes for that patient do not remain in the system.

Consideration should also be given to the record selection functionality to ensure that delete messages are not set to VINAH due to records being archived.

Ephemeral records

Where records are created in local systems but are not persisted due to their entry being in error or similar reasons, these records will not be assembled using the transaction checkpoint method.

Non record-level errors

The logical data structures outlined in the transaction checkpoint method do not outline a mechanism to handle errors that are returned from VINAH at a file level. These errors should be attached to the submission file record in the local system and made available for the user to view and action where appropriate.

Submitted corrected rejections

The process for selecting records from the transaction log covers the re-submission of corrected records; the action of correcting a record will set the last local update date to a date greater than the last submission date resulting in the corrected records being included in the next record.

It is acceptable to resubmit records that were previously rejected even though they have not changed, with the expectation that the records will again receive a rejection.

Resending previously accepted records

In the normal VINAH lifecycle, messages are created in response to user triggers such as changing the value on a clinical record.

In some cases it may be necessary for a user to resend a record even though it has not changed. An example of this would be an existing record that needs to exist in the system as it is referenced by another record.
Another scenario is if a data quality investigation found previously undetected errors in previously submitted records, those records could be extracted and re-sent to VINAH in order to be re-validated. In this case, the software designer may include functionality to allow users to flag that a record or set of records should be sent to VINAH, without requiring a change to be made to the record.

**Record locking and concurrency issues**

This method avoids some of the issues relating to record locking and concurrency, as the checkpoint date/time provides a fine-grain fixed boundary around which to select data. Local database transactions can continue to occur after the checkpoint date, even if the checkpoint procedure is still running; it is conceivable that the system should be able to remain accessible during this period.

**Patient merges**

In addition to the data selection logic outline in the checkpoint procedure, further logic will have to be applied in the case of a patient merge. As the Patient Identifier is effectively changed on all the clinical records types, the vinah_record_transaction structure will likely need to be updated to ensure that the same record_id is recorded. This will ensure that the altered records are not re-flagged as inserts or deletes to VINAH.
Method 2 - Message/Transaction queuing

This method involves generating transaction messages on the fly and queuing them until submission. The advantages of this method include attaching trigger mechanisms to either entry screens or database structures to assemble VINAH messages as the local transactions occur.

There are several downsides to this method which would make implementation highly unlikely to be successful and in some cases near impossible:

- It is extremely likely to get out of synch and correcting such scenarios is difficult; one or more messages in the queue may be rejected for data quality reasons, and the stream could then be broken due to the referential integrity requirements of VINAH.
- Where ephemeral records are frequently created, for example records entered in error and deleted shortly after or temporary records, the message queuing approach will most likely pick up these transactions and submit them to VINAH; this would create unnecessary overhead on the VINAH processing lifecycle.
- Many data entry screens would generate update transactions to records as they are progressively built up during data entry. In many cases these updates are irrelevant to the VINAH data collection, and are not necessary to submit. Excessive update transactions would create unnecessary overhead on the VINAH processing lifecycle.
- In many systems, the conceptual VINAH record is built up of data from various screens or physical database structure. Attempting to assemble a VINAH transaction from various locations as a result of a data entry trigger may prove problematic.
Method 3 - HL7 message interception

This method is similar to the message/transaction queuing. It involves effectively listening to an internal HL7 message stream to the VINAH system. The advantages of this method include being able to plug into an existing stream of transaction messages which could make implementation of an existing system quite simple.

There are several downsides to this method which would make implementation highly unlikely to be successful and in some cases near impossible:

- In many cases, VINAH enforces different validation rules than those provided by both the HL7 standard and local systems.
- Where HL7 messages are sent between local systems, these messages are usually accepted as-is; data may be invalid from a business point of view however systems generally do not enforce rules at this level.
- VINAH data must meet the business rules as set out in Section 4 – messages are not accepted unless they are valid in order to preserve data quality of the entire collection.
- Given that VINAH may not accept a particular message for any given reason, the stream of data may effectively be broken and transactions will be out of order or synchronisation.
- Some standard HL7 messages do not carry the data elements as required by the VINAH collection; any intercepted messages would have to be at the very least transformed to meet the VINAH message profiles and data element definitions/code sets; in the worst case would unusable.
- In many systems, the conceptual VINAH record is built up of data from various screens or physical database structure. Attempting to assemble a VINAH transaction from various locations as a result of a data entry trigger may prove problematic.
- VINAH is a statutory data collection which relies on enforcing data quality rules to ensure that decisions can be made on accurate data. This principle is at odds with the free-flowing nature of automated HL7 which enforces little or no business rules in order to guarantee the delivery of messages.
Data submission lifecycle

Overview

The submission process lifecycle for the VINAH MDS data transmissions is illustrated in the following diagram.

VINAH Submission Process

<table>
<thead>
<tr>
<th>Health Care Organisation</th>
<th>HealthCollect Portal</th>
<th>VINAH Validation Engine</th>
<th>VINAH Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate VINAH Extract</td>
<td>Acknowledgement File (ack.txt)</td>
<td>Validate Submission</td>
<td>VINAH Repository</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Validate Submission</td>
<td>VINAH Validation Register</td>
</tr>
<tr>
<td>Correct Errors</td>
<td></td>
<td>Commit Valid Batches</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reject Invalid Batches</td>
<td></td>
</tr>
</tbody>
</table>

Queries on Portal or VINAH processing system availability, for example in the event of non-receipt of message acknowledgment after a reasonable time, may be directed to the HDSS Helpdesk via the email address hdss.helpdesk@dhhs.vic.gov.au or via phone during business hours on (03) 9096 8595.

Unlike other health data collections, a given VINAH submission need not contain all data for a particular month; nor is it restricted to only data for a given month. A VINAH submission can contain one or more individual data records, represented as HL7 messages.

In VINAH, monthly submission requirements are assessed by analysing all data submitted at a point in time. By the due date it is expected that all data pertaining to the relevant period has at some point in time before the due date, been submitted. This data may indeed have been submitted in one large file, or may have come in a series of smaller files, or indeed on a message-by-message basis.
**File transmission automation**

Once the submission package(s) are compiled, the files should be transmitted to the HealthCollect Portal – See Section 5d – *HealthCollect Portal manual transmission process*. Requiring users to manually upload files to the HealthCollect Portal user interface is acceptable but discouraged. It is strongly recommended that the function of submitting VINAH files and receiving reports be built into the submitting software using the VINAH web services. This will eliminate the need for users to manually upload and download reports. This also presents the opportunity for the system to perform submissions in the background, and only present errors to users where actions are required.

**HealthCollect Portal (user interface)**

Advantages:
- Dump files to disk, let users handle the process - little or no implementation required

Disadvantages:
- Possibility of uploading the wrong or incorrect version; incomplete file uploads or other user error issues
- Burden on users to manually handle files
- Users must log in frequently to check for submission reports
- Weakened auditing capability by the sending system
- Each user of the system has to gain proficiency in the manual transfer process, creating support overhead when new users or systems introduced out to new site.

**HealthCollect web services**

Advantages:
- Allows systematic control, auditing and logging of transmissions
- Allows automatic retrieval of submission reports, saving users time
- No manual handing of files avoids user error problems, reducing support calls
- Allows for the possibility of automatic submissions

Disadvantages:
- Implementation required

**Uploading multiple files**

Where multiple files are being uploaded, files will be run in order of the date/time of upload/submission. Files should not be uploaded all at once using the website as they are all assigned the same upload date, and there is no guarantee that files will be processed in order of file name. Instead, files should be uploaded one at a time, i.e. spaced by a period of at least a second).

**Submission file naming**

The file naming convention to be used when submitting VINAH files is as follows:

The file name is defined and tested by the following regular expression:

\[0-9_a-zA-Z]{1,30}\.[hl7|xml|zip]\)

- The file extension should be .hl7 for an HL7 file,
- The file extension should be .xml for an XML file,
- A valid organisation identifier must be the first characters in the file name,
- The organisation identifier must exist in the code table HL70362,
- The file name must be unique in time. File names may only be re-used if the original file was not acknowledged by the HealthCollect Portal,
• It is highly recommended that the date and time the file is created by the health service be used to form the file name. This ensures that unique file names are always submitted,

• It is recommended that the time period of data in the file is not used as part of the file name, as this may not generate unique file names.

Example: hs_20100601_01.hl7

In the example, the organisation identifier is the first two characters (‘hs’) and the date of data extraction (01 June 2016) has been used along with a sequence number (‘01’) to provide a unique file name. The structure of the file name should reflect the time of generation of the file and should not attempt to reflect a time period of the data contained within. It is acceptable to include in the file name other metadata such as the system or application that generated the file to avoid the possibility that two different systems at the same health service produce the same file name. It is important that the system that generates the VINAH submission file also generates the file name. Users should be instructed not to alter the file name unless instructed otherwise.

File names must be unique for each submission across the life of the data collection. A file name must never be reused if it has been received by the VINAH system. This holds even if the file is empty, corrupt, contains numerous errors and is subsequently resubmitted, or the file has been purged and the same data is to be re-submitted.

Submission file persistence

Although it is possible that a submission may be made directly from an application to the HealthCollect web service without writing a physical file to disk, for auditing purposes it is recommended that a copy of each submission file is persisted to a local or shared disk.

Submission authorisation

In addition to receiving a HealthCollect Portal username, each username must also be registered to submit VINAH data for a particular organisation identifier. In addition, organisations must be registered to submit VINAH data for a particular program. These registrations can be undertaken by contacting the HDSS Helpdesk.

Character encoding

Data must be transmitted as a 7-bit ASCII encoded file.

File sizes

Submission file sizes shall be no greater than 5MB.

Submission files should contain no more than 5000 records/messages.

The ideal file size providing optimum processing speeds is between 2-5MB.

If the need arises to submit more than this amount (e.g. if submitting historical data) then the messages should be spread across multiple files.

VINAH developers should include provisions to output data across multiple files based on a configurable threshold number of messages per file. Consideration should be given to the fact that one submission report is returned for each file.
VINAH submission reports

Each submission is responded to in two ways:

- Upon submission of the file and receipt by the VINAH system, a text file ([filename].ack.txt) is returned to the sending user’s HealthCollect Portal account. This file indicates that the file has been received by the system and has been placed in a queue for processing. It also contains information on how long the system expects to take to process the file with respect to the amount of data ahead of it in the queue and the amount of data in the submission file itself. The acknowledgement is further documented in Section 7.

- Once the submission file has been processed, an XML submission report file is returned to the sending user’s Portal account. This file summarises the processing operation, lists all validation events that occurred during the processing, and provides an acceptance status for each batch and message submitted. The size of this submission report can in some cases be proportional to the size of the submitted file, especially where large numbers of errors are resultant. The submission report is further documented in Section 7.

Manual v automatic submission/acknowledgement

The following sections outline some of the best practices to consume the submission report. Although at the start of the VINAH development cycle many vendors opt for manual uploads and downloads, it should be noted that this cycle is expensive time-wise.

It is strongly recommended that the function of submitting VINAH files and receiving reports be built into the submitting software using the HealthCollect Portal web services. This will eliminate the need for users to manually upload and download reports. This also presents the opportunity for the system to perform submissions in the background, and only present errors to users where actions are required. For more information regarding the use of the VINAH web services, please contact the HDSS Helpdesk.

Consuming the submission report

Although the submission report provides information for the sending user around invalid data that needs to be corrected, it is important to note that validation messages in the submission report are only a part of the overall process.

The submission report contains important flags which need to be captured in the sending system’s VINAH transaction log to ensure ongoing synchronicity. As such, it is not a viable approach to expect the sending user to interrogate the submission report file to investigate errors and control the submission process manually. The report does not contain identifying information for each record apart from the key for that record. Unless the key can enable the user to easily locate the record in error, using the XML report in isolation to identify records will be difficult for a user.

The recommended approach is to implement a process in the sending system’s VINAH component which can consume the returned submission report. The XML in the report is easily readable by a machine and implementing this process through the software is far more reliable than any manual process involving the user, and allows the sending system to manage the VINAH submissions in a structured way.
Consumption process

The consumption process can be summarised using the following narrative, with reference to the transaction checkpoint method:

1. Provide a function to allow the user or automatic process to import the XML submission report into the VINAH submission generation software. The optimal model is to implement a service at the user end to automatically poll the HealthCollect Portal (via the web services) for submission reports.

2. Process the `<Acceptance>` node and for each batch and message set the value of these data elements in the VINAH transaction log:

   vinah_transaction
   - File name - <submission\filename>
   - ack_datetime - <submission\process_end_date>

   vinah_record_transaction
   Fields such as transaction_type, vinah_entity etc. can be inferred by linking to the mcid.
   - mcid - <acceptance\batch[n]\message[n]\mcid>
   - rejection_msg - <validations\validation[n]\edit_text>
   - accepted_flag - <acceptance\batch[n]\message[n]\accepted>

3. Collate errors listed in the `<Validations>` node in a manner that suits the users and the local software. It is recommended to summarise the errors by error code and by patient, and provide links to screens where local administrative staff can click through easily to correct errors. It is also recommended that information that will be useful to identify records be included, such as Contact date/time, Referral received date, Episode start date/time, etc.

4. Where a `<Validation>` node exists without a `<mcid>` value, the error exists at a file or batch level. These errors should be attached or associated to metadata about the individual transaction file (this may be more than one file), and available for users to view where appropriate.

Waiting for a submission report

Once the VINAH submission has been packaged and exported and/or uploaded, the sending system is effectively in a ‘waiting’ state and expects to be able consume the submission report to complete the submission process and flag the appropriate records.

Software designers may wish to build in the ability to rollback this state where a submission report is not or cannot be received. This will avoid situations where the sending system is waiting for a report that it can never receive, or for a set of data generated in error that has been retracted from the Department.

Persisting messages

The validations, or rejection messages generated by VINAH, are generally associated with the version of a record in the local system that was submitted. It should be noted that a record may be changed by a user in the local system between the times that the submission is made and the VINAH rejection received.

The rejection should stand in the system until such time that the record is subject to either an update or a delete and a user of the system has marked the rejection as repaired. At this juncture the VINAH rejection is deemed to be no longer applicable to the record.

Non record level errors

Several classes of VINAH errors do not relate to a specific record in the system, rather a set of records, the entire submission file or other data quality scenarios. These errors messages, where repairable by the user, should be presented to them and an option included to flag the error as repaired.
Errors that users cannot repair

There are several VINAH errors that aim to preserve the integrity of the data relative to both the submission itself and also previously submitted data. Much of the referential integrity class of VINAH errors are not data quality errors, rather, they reflect errors in the submission process, the logic used to extract and package VINAH data or the process used to maintain a VINAH Transaction Log.

As a rule, when non-user repairable errors occur, this should place the system in a state of exception and require support. These errors should not be presented to the user as they are confusing and not errors that they have control over. The VINAH submission system may intercept these classes of errors and respond to them with predefined processes if necessary.

Where a System, Process or File level message has been received, all records should be flagged as not accepted, but not invalid.

Purging submissions

There will be occasions when a series of submissions to VINAH will need to be cleared or purged so that a fresh set of data can be submitted. This is a common case where an upgrade is taking place or a systematic error has been detected.

There should be no reliance on using the purge or purge-after-load process in order to submit regular data. If a file contains validation errors, a purge should not need to take place in order to submit a corrected file. This creates significant overhead at both ends of the process. Effort should be assigned to ensuring that a VINAH transaction log is developed to avoid the need for regular purging and bulk loading.

Where a major synchronisation issue has occurred, purge request files should be sent in order to purge back to a certain point in time. These purge request files can easily be created from the VINAH submission report files.

Manual data purges will only be undertaken by the Department where the history of submission has been lost or corrupted and the sending system cannot request the purges.

The procedure to generate purge request files is documented in Section 5e – Submission Purge Procedure.
Implementation considerations

Keeping track of transmissions

The first time a new entity (e.g.: Patient/Client, Episode, etc.) is successfully sent to VINAH, it is created with an Add message. Subsequent changes to that entity must be sent with an Update message. Further transmission of an Add message once the first has been accepted will reject, as will transmission of an Update without first sending an Add. For this reason, software systems transmitting to VINAH will need to keep track of what information has been sent in order to know which type of message to send next time around.

Once processed by the VINAH validation engine the XML submission report may be collected from the HealthCollect Portal. Organisations' software systems should have a method of reading this information to determine which messages have been successfully processed and which need to be resent. For example, if a Patient/Client Add message is transmitted and succeeds and the Patient/Client's details are subsequently changed an Update message must be sent next time. However, if the original Add failed, the Add message must be re-transmitted, either with the updated details included or as originally sent and followed by an Update.

The transaction checkpoint method outlined in this document outlines practical approaches to manage the submission lifecycle.

User data amendments

In general once software development and implementation issues are resolved, the bulk of VINAH validations that are triggered will be data-related, that is, data entered into the source system that violates a VINAH business rule or validation. Ideally the source systems will identify these instances prior to transmission, however in the event that processing does generate errors of this type, the source system must provide a method for users to access and correct the information and then re-transmit the information to VINAH.

Correction of the information should be through the source system's user interface and must be applied to the source data and not the HL7 extract or other extract database. Users should be able to identify the information in need of updating from the edit report and access that information for amendment.

Identifier schemes

Clinical systems often undergo reengineering which affects the way in which identifiers are issued (e.g. patient identifiers/unit record numbers etc.). Consideration should be given to the effect of changing identifier schemes; these changes should not cause integrity issues nor should they affect the ability of related records to be reconciled.

Program/Stream changes

In order to preserve the integrity of the VINAH repository and ensure the appropriate validation logic is applied to all data, the changing of the Program/Stream value on an Episode is restricted. This means that once an Episode has been inserted, the Program/Stream data element value cannot be changed through an update message.

If an episode program/stream needs to change, all affected episodes should be deleted and re-inserted.
Multiple source systems

While the Department recommends use of a single source system for all VINAH transmissions within an organisation, it will not always be possible to achieve this in practice. Accordingly, many of the significant changes to the VINAH specification for 2007-08 were implemented to facilitate reporting from multiple systems - particularly the shift from derived to explicit episode reporting.

- Further changes in 2012-13 allow for organisations to use the same Patient identifiers, referral identifiers, episode identifiers and contact identifiers across multiple systems for different patients.
- The Local Identifier Assigning Authority (LAA) reported with the Patient identifier has been applied across all identifiers when determining whether an identifier is unique. This relies on LAA values being unique to a vendor system within an organisation.
- Submission timelines apply across all VINAH data.

Organisations must manage the logistics of separate system submissions to the HealthCollect Portal.

Front end validations v VINAH validations

Software vendors are encouraged to implement validations in their systems to enforce the appropriate business rules, VINAH or otherwise, at the point of entry. This will reduce the amount of validations triggered by VINAH.

It should be noted that while exercises to map VINAH business rules and validations into a user-facing application are encouraged, such an effort should not be deemed to have eliminated the possibility of an error being returned from VINAH; systems need to have the capability to deal with any error returned from VINAH even where a front end validation was designed to capture that error.

As outlined in Section 8, if one or more validations are triggered at in a certain group, the processing for that level will discontinue. This means that certain validations will not be applied to data at a certain point in time, but this does not mean that the validation will not be triggered in the future.

Therefore any attempts to reconcile local validations and VINAH validations may result in inconsistencies due the fact that rules may be applied at different points in time.