JCQ[CIC] A2C Data Standards Specification

Section 9

Feedback

2022 Version

28th February 2022

Table of Contents

1 Introduction

The concept of electronic message receipt and processing responses does not exist in EDI. The sender has to assume the file was received and processed successfully unless they hear otherwise: this can lead to issues later if it turns out there is a problem that was not flagged and/or fully resolved. The aim in A2C is to respond immediately to confirm receipt of the message, followed by feedback to either advise of any issues with the data or to confirm the message, and/or all the transactions within it, were processed successfully.

Feedback is a response to a data message received from a trusted party; which is generated after inspecting the business data within the message. The response can be at a technical level where the receiver wants to convey any issues with processing the message, such as invalid data; or at a business level where it confirms either successful consumption of the message or highlights issues with data received such as errors, missing data, conflicting data and data not conforming to business rules. For further information on feedback refer to:

- the stated Service Level Recommendations (SLRs) for providing feedback, within the relevant sections of this Spec. These typically vary from one to seven working days depending on the transaction type.
- The 'Action Codes' sheet of Appendix 3 which defines each feedback message type corresponding to a data message containing a particular transaction type. Feedback type messages are specified in bold italics.
- Appendix 5 which lists the different feedback codes and descriptions which can be used in a feedback message to highlight any issues or confirm the successful processing of data.
- XML Schema Usage in Section11: *Solutions Architecture* for some definitions and rules around XML messages.

A feedback message is a response to a message received; the attribute 'Ref_Message_Id' in the 'FeedbackMessage' block in the XSD provides the link to the original message and is mandatory for all feedback messages, with two exceptions:

1. Feedback code 0003 – 'Messages out of sequence'
   As described in Section 11: *Solutions Architecture*, this is a special type of message and not a feedback message; hence Ref_Message_Id is not populated.
2. Feedback code 0004 – 'Message structure does not conform to the A2C schema'
   The behaviour of this message is also documented in Section 11: *Solutions Architecture*. Due to the nature of the problem, it may not be possible to retrieve the original message identifier, hence Ref_Message_Id is not required.

2 Message Processing at Receiving End

Message processing is performed by the Message Processing Component at the receiving end. The message goes through the following processes:

1. Different stages of validation
2. Generation of feedback for the original sender
3. Committing the data into the receiving system's database.

2.1 Staged Validation of a Message

The validation process is broken down into four sequential stages, T1 to T4. Each stage must follow the other and each is responsible for validation steps to an agreed criterion within time limits as per the SLA. All feedback messages created at any stage of validation will be routed back to the sender using the ebms:From->PartyId:

Validation Stage	Validating Component	Behaviour	Response Type
T1. Transport Stage	Transport Component	On failure must not proceed to next stage.	ebXML errors eg EBMS:0006 Empty Message Partition Channel or EBMS:0303 Decompression Error
T2. Payload Stage	Message Processing Component	On failure must not proceed to next stage.	A2C feedback messages
T3. Data Stage	Message Processing Component	On failure can proceed to next stage.	A2C feedback messages
T4. Follow-on Validation Stage	Other (Implementation Specific) eg ULN LRS call and validation or workflow.		A2C feedback messages

Table 1 Message Validation Stages

Message Processing Component is the component within the receiving system which is responsible for processing the messages after it is received by the transport.

2.1.1 T1: Transport Validation Stage

Transport validation is executed by the Transport Component and is responsible for all non-payload related errors. The full details of the Transport Validation Behaviour and Error tables can be found in Section 6 Error Handling of the ebMS 3.0 Core Specification. See also Section 12: A2C Transport Specification, heading 7.4 Support for ebMS processing errors.

2.1.2 T2: Payload Validation Stage

Payload validation stage is responsible for ensuring the validity of the payload as a whole. If a failure occurs feedback is constructed and transmitted and no further stages are executed. These steps can be followed in the sequence below:

Validation Step	Validation Error Response
1. The payload is unreadable, or corrupt, or the message is invalid against the A2C schema	0004 Message structure does not conform to A2C schema
2. The message does not refer to one of the defined transaction types or the action code in transport (from transport metadata) does not match with the TransactionName attribute in the message header	0007 Message rejected: TransactionName not recognised
3. The Initiator->Party->Party_Id within the message header does not match the ebms:From->PartyId within the transport metadata.	0005 Message rejected: Message initiator not recognised
4. The Receiver->Party->Party_Id within the message header does not match the ebms:To->PartyId within the transport metadata	0006 Message rejected: Message receiver not recognised
5. The message is out of sequence. (See Message Sequencing in Section 14 *Solutions Architecture*)	0003 Messages out of sequence
6. The message does not contain all the mandatory data blocks for the transaction type	0008 Message rejected: missing or invalid data blocks
7. The message contains data blocks not defined for the transaction type	0008 Message rejected: missing or invalid data blocks
8. The message sequence number was already used by another message received from the same sender earlier	0009 Message ignored: A message with same sequence number already received earlier
9. A message is received by an awarding organisation from a centre with a transaction type other than Request Product Catalogue or Centre Set-up Notification, before receiving Centre Set-up Notification	0010 Message rejected: Message initiator not recognised as A2C compliant
10. Software company name, package name or package version is missing in a message from a centre to the awarding organisation	This will be an advisory rather than an error message. The message will still be processed and will not be rejected due to this missing information. The advisories applicable are 0011, 0012, 0013
11. A Centre Set-up Notification is received again from the same centre	0014 Centre Set-up Notification transaction cannot be submitted as this process has already been completed successfully

Validation Step	Validation Error Response
12. Awarding organisation receives a message with a valid transaction type as specified in Appendix 3, but the transaction type is not supported by the awarding organisation	0202 Prohibited transaction: <Transaction Type/ Action Code> not supported by this awarding organisation
13. Exchange Name is not 'JCQ-A2C'	0017 Exchange Name <Exchange_Name> is not supported. Supported Exchange names are: <Supported Exchange_Names>
14. Schema Version or Exchange Spec Version is not supported	0016 Schema/Specification version <SchemaVersion> / <ExchangeSpec_Version> is not supported
15. Centre Party ID used for a transaction does not match the initiator party Id. These are required to match as sender identity assurance is performed against the initiator party Id.	0019 Centre Party ID for Transaction <Transaction_Centre_Party_Id> does not match Party ID used in transport <Transport_From_Party_Id> 0002 - Message rejected

Refer to Appendix 5 for further details of the feedback codes.

2.1.3 T3: Data Stage

Refer to Section 11 *Solutions Architecture* for definitions of record and transaction.

The data stage validation is responsible for the detailed validation including:

1 Attribute level validations including data format, data type, mandatory check and check for harmonised values as specified in Appendix 2. Applicable attribute level feedback codes specified in Appendix 5 should also be considered.

2 Transaction level validations such as:

- a Check for absence of an attribute or the whole record in a data block which may invalidate the transaction due to business rule violation, example scenarios include:
  - i ULN is required for a Named Order, a record for ULN exists in PartyRelationshipRole, but ULN provided is empty
  - ii ULN is required for a Named Order but a record for ULN does not exist in PartyRelationshipRole data block
  - iii in a Named Order if a related record does not exist in QEBooking data block against a transaction identifying record in QELearnerBooking data block
- b Check for business rule violations as defined in the business process specifications
- c Check for duplicate transactions in case the same transaction is already provided earlier or it is duplicate within the message
- d Consistency checks such as an amendment is received before receiving the original transaction.

It is suggested that the transactions are processed from top to bottom as provided within the primary data block.

2.1.4 T4: Follow-on Validation Stage

This process starts after the first level of validation is sent. The follow-on validation stage is responsible for any long-running or externally dependant or non-automated validation checks as defined by the business process and feedback will be constructed and transmitted within the defined SLA.

Step	Feedback Message Types
Any automated validation which cannot be carried out during T3 synchronously such as ULN verification with LRS.	All applicable feedback codes (see Appendix 5). Example: 0133 ULN failed validation with LRS (see Example F on page 22)

2.2 Method of Data Validation

Data is received in an XML message which is constructed as blocks of data, each containing related records.

Implementers can choose to construct separate data objects for each transaction consisting of one record from the primary data block which identifies the transaction and one or more related records from other data blocks; and then apply validation rules to each transaction separately. This is a reasonable approach where it becomes easy to accumulate attribute level feedbacks within a transaction, apply business rules on the whole transaction, and then create transaction level feedbacks.

Implementers could apply all general attribute level validations in all data blocks first, and then perform transaction level validations. In this case all the feedbacks accumulated would have to be classified under different transactions before constructing the XML feedback message.

There are other possible validation methods, but implementers should ensure that the data is successfully consumed or rejected after applying relevant business rules and appropriate feedback message is generated.

3 Definitions

Each feedback must have a level and severity:

Level: Level defines the scope of data the feedback is applicable to and there are three levels of feedback:

1. Attribute – Feedback is related to a single attribute or a combination of attributes within a single record in a data block. Attribute level feedback is generally provided if the data is invalid or violates business rules.
2. Transaction – Feedback is related to a single transaction. Transaction level feedback of severity type 'Error' is the result of one or more attributes in error or if the specified behaviour of the transaction is not followed. Feedback on a transaction of severity type 'Confirmation' is given if the whole transaction is successfully processed.
3. Message – Feedback is related to the entire message or all of the transactions within the message.

Severity: Describes the severity of feedback and indicates whether records have been accepted by the receiver and whether any action is required of the sender. Note that the full natural language values of Warning, Confirmation, Advisory and Error are provided in the feedback messages.

1. Advisory: Feedback is for advice only and no action is required of the sender. The data is accepted by the receiver.
2. Warning: Feedback should be acted upon by the sender. The data is not critical at this point of time. The sender would need to send valid data in order to enable further business processes to be completed successfully.
3. Error: Feedback must be acted upon by the sender. The data is required by the receiver for the business processes and it will not be accepted until the errors are fixed.
4. Confirmation: The data has been accepted by the receiver.

Level	Severity	Behaviour
Message	Advisory	This is for information only. No action is required by the sender. This feedback does not indicate the acceptance or rejection of the transactions within the message.
	Warning	Action should be taken by the sender to ensure that all the information in the message has been stored by the receiving system. This feedback does not indicate the acceptance or rejection of the transactions within the message.
	Error	Action must be taken by the sender. No transactions are processed by the receiving system.
	Confirmation	No action required by the sender. All transactions are accepted and stored by the receiving system.
Transaction	Advisory	This is for information only. No action required by the sender. This feedback does not indicate acceptance or rejection of the transaction.
	Warning	Action should be taken by the sender. Although the transaction is accepted and stored by the receiving system, the receiver will need valid data to enable completion of further business processes.
	Error	Action must be taken by the sender. The transaction is rejected by the receiving system.
	Confirmation	No action required by the sender. The transaction is accepted and stored by the receiving system.
Attribute	Advisory	No action required by the sender.
	Warning	Action should be taken by the sender. The data is not critical at this point of time, but would be required by the receiver for successful completion of further business processes.
	Error	Action must be taken by the sender. One or more attribute level errors may result in a transaction level error.
	Confirmation	No action required by the sender.

Table 2 Levels and Severity for Feedback Codes

A list of feedback codes, the level it applies to and severity is specified in Appendix 5.

The following levels of feedback are **not** used in A2C:

- Record Level feedback: There is no concept of a record fully accepted or rejected. A record is considered to be accepted if the corresponding transaction is accepted.
- Data block level feedback: It is never reported whether a data block is fully accepted or fully rejected.

4 Feedback Behaviour

1. A feedback message is identified by the TransactionName in the message header. The message uses the FeedbackMessage block of the schema and the Ref_Message_Id is mandatory as it signifies the message identifier of the original message against which the feedback is given (except for messages 0003 and 0004 – see Section 11: Solutions Architecture for further details on handling Out of Sequence and Non Conforming Messages)
2. There can be multiple feedbacks at message, transaction and attribute levels. More than one feedback code can be provided at the same level to clarify different issues associated with data.
3. A feedback message may contain:
   a Message level feedback(s) only
   b Message level feedback(s) accompanied by transaction level feedback(s)[1]
   c Transaction level feedback(s) only
   d Transaction level feedback(s) accompanied by attribute level feedback(s)
4. Each feedback has two parts; the feedback code and a feedback text. Some feedback texts are template based where values are embedded into the message while creating the feedback message dynamically. This is done to make the feedback text more user-friendly and informative.

   Appendix 5 contains feedback codes with a description and some notes describing the feedback. Template based messages must not be used for missing attributes. There should be different feedback codes for different attributes, which will enable the feedback processing system to automatically determine the missing attribute.
5. Each feedback message must relate to only one received message referred to by specifying the MessageId of the original message in the Ref_Message_Id attribute of the 'FeedbackMessage' block in the schema.
6. Multiple feedback messages may be sent against a single received message. For example, if a message contains 10 transactions, then one feedback message can contain feedbacks for the first 4 transactions and a second feedback message can contain feedbacks for the other 6 transactions.
7. Feedback against a particular transaction is not sent multiple times unless the business process defines a Follow-on Validation Stage (see T4 above). For example, in the case of a Named Order, the ULN validation occurs after the transaction is accepted and a separate feedback against the same transaction is sent after the ULN is verified with the LRS database.
8. If feedback is already sent on a transaction, but further issues are identified later on then manual action must be taken by the receiver of the transaction to settle the issue with the sender.
9. Any follow-on validation can result in feedback related to business rules pertaining to follow-on validation only and must not raise feedback on other aspects of that transaction which could not be identified at T3 validation. For example, follow-on validation against a Named Order should contain feedback codes such as '0133 – ULN failed validation with LRS'. The

[1] This combination is only allowed if all message level feedback is of advisory severity. Implementers may decide whether the message level advisory feedback should be provided with just the first transaction or included with all transactions.

follow-on validation must not raise feedbacks like '0107 – Date of Birth missing', which could not be identified at T3 validation.

10. Duplicate transactions:

    If a message contains two or more duplicate transactions, then the message processing component will process the first transaction and reject the rest of the duplicate transactions. The duplicate transactions are identified using the identifiers in the primary data block.

    If one or more transactions within a message are duplicates with a transaction data already provided in a previous message and is stored in the receiving system, then the message processing component will reject all these duplicate transactions.

    Note: There is a related scenario pertaining to Named Orders, documented below under 'Behaviour related to Named Orders'.

11. If an attribute is validated successfully, an attribute level confirmation will not normally be used, instead attribute level feedback is included by exception only. In cases where a problem, or a potential problem, is identified, an attribute level feedback with severity advisory or warning or error is used. For the few cases where it may be helpful to provide attribute level confirmation, feedback codes are included in Appendix 5. It is not obligatory to provide such confirmation; it is at the discretion of the receiving party. For example, one awarding organisation may choose to provide confirmation of successful ULN validation, whereas another may only provide feedback on problems with ULN validation.
12. If an attribute is invalid but it is not required to complete the transaction, either now or later, and therefore no further action is essential, then Advisory severity is used for the attribute level feedback.
13. If an attribute is invalid but it is not required immediately to complete the transaction, an attribute level warning is used.
14. If an attribute is required to complete the transaction, but the value is invalid, then Error severity is used at attribute level.
15. An attribute level feedback will always appear under a transaction level feedback to make it hierarchical within a feedback message.
16. If one or more attribute level feedbacks are provided, then there must be at least one transaction level feedback provided for that transaction within which the attribute resides.
17. If one record in a data block is part of more than one transaction, and attribute level feedback is provided for one or more attributes within that record under a particular transaction, then the same feedback must be repeated for other transactions as well if it is applicable. This will clearly justify the transaction level feedbacks under each transaction. For example, in a message with Named Orders one learner relates to two bookings, and the learner UCI is found to have been already used for another learner causing a conflict. In this case feedback 0143 (Supplied UCI of 12345A already in use by another learner) is provided against Relationship_Reference under both the transactions.
18. When an attribute level feedback is provided, the attribute is uniquely identified by specifying the identifiers of the record in which the attribute is present. The feedback contains the name of the attribute, the existing value of the attribute, the feedback code, severity and feedback text.
19. Usage of feedback code 0001 at message level:

    If all the transactions in a message are successfully processed with no warnings or advisories at the transaction level or attribute level, then transaction level feedbacks with

confirmation severity are not provided, rather one message level confirmation with feedback code 0001 is provided (see Example D below).

- If feedback for all the transactions in a message is not sent in one message (See 6 above), then feedback code 0001 would not be used.
- Feedback code 0001 is not sent as a result of follow-on validation (T4). For example, if a received message contains ten Named Orders, two of them have learner details containing ULN, then follow-on validation occurs to validate these two ULNs. The follow-on feedback is sent to the centre after the validity of these two ULNs is received from LRS. This follow-on feedback cannot contain 0001 as it is reporting issues with ULN attribute only and not all the ten transactions originally received.
- As indicated this feedback code is only used when all the transactions in the message have no errors, no warnings and no advisories and not in any other scenario.
- In a scenario where all the transactions are processed successfully with no warnings or advisories or errors at transaction or attribute level, then instead of providing one feedback per transaction, only one message level feedback 0001 should be used to create a smaller size message. This is preferred but not mandatory – see Example E for a description of an alternative approach.

In summary:

i. Message level feedback code 0001 cannot be combined with any transaction level or attribute level feedback.

ii. Message level feedback code 0001 can be combined with message level advisories. For example, a feedback message can consist of 0001 and one or more of 0011, 0012 or 0013.

20. Where feedback includes confirmation that a transaction has been successfully processed (either '0200 <Transaction Type> successfully processed' or '0203 <Transaction Type> (<QE_Booking_Type>) successfully processed'), this means that the transaction has been accepted and registered in the receiving system. Confirmation that a transaction has been successfully processed does not guarantee that all business processes related to the order can be successfully concluded. Any Warning messages relating to attributes within a Confirmed transaction will still require action from the original sender. Failure to act on resolution of Warnings may mean that business processes related to the transaction cannot be successfully concluded, although it will not affect acceptance of the transaction.
21. If one or more transactions in a message are rejected or accepted with warnings then transaction level feedbacks accompanied by any relevant attribute level feedbacks are always provided.
22. If all the transactions within a message are rejected due to business rules violation, then feedback on each transaction along with any associated attribute level feedback is provided. This will not be associated with message level feedback '0002 – Message rejected'.
23. Some feedback codes permit variable severity. This allows different systems to apply different business rules and apply the appropriate severity level. For example, an awarding organisation that requires a UCI for Orders may provide Error severity feedback if it is not supplied. In contrast, another awarding organisation that does not require a UCI may provide Advisory severity feedback.
24. If a transaction is rejected, then the transaction identifying record in the primary data block and all the records in related data blocks belonging to that transaction are also rejected. However, records in the related data blocks might be accepted as part of another

transaction. Example scenario: there are two bookings for the same new learner for different qualifications in one message. The first transaction is rejected due to a business rule validation failure, but the second transaction is successfully processed. Therefore, the learner details are successfully accepted as part of the second transaction and need not be provided again by the centre after fixing the issues with the first transaction.

25. Handling missing data:

    a Using transaction level feedback for attributes that are completely missing from the incoming message XML.

    For required attribute(s) that are not present in the message XML, feedback will be provided at transaction level. It is not possible to feed back at attribute level on an attribute that does not exist. There is a specific feedback code for each attribute that could be mandatory, permitting unambiguous identification of the missing attribute(s).

    In other words, if an attribute is required to complete the transaction, but either the record containing that attribute is completely missing in the relevant data block, or the attribute is missed from the record, then feedback is not given at attribute level, rather feedback is given at transaction level.

    For example, in a Named Order where the PartyRelationshipRole record is present and Party_RR_Reference_Type is populated with the value 'ULN', but the Relationship_Reference attribute is not provided, then the feedback code '0145 – ULN missing' should be provided at transaction level.

    b Using transaction level feedback for attributes that are left blank in the incoming message XML.

    For required attribute(s) that are present but not populated (ie left blank) in the message XML, feedback will be provided at transaction level. It is not possible to feed back at attribute level on null values for an attribute. See point (a) above.

    c Using transaction level referential integrity feedback.

    If one or more records are missing in a related data block which does not allow the transaction to be processed successfully and no specific business rule violation can be identified due to inconsistent data, then missing attribute feedback should not be used; instead a referential integrity feedback is generated against the data block record which refers to the missing record.

26. Feedback on feedback

    If a feedback message does not conform to the A2C schema, the receiver may send a Non-Conforming Message Notification (NCMN).

    Assuming the feedback message conforms to the schema, no further feedback is allowed on feedback messages.

    Rule R8 of section 12.1 of the Solutions Architecture states: "If an invalid feedback message is received by the original sending system (eg feedback message contains erroneous feedback codes) then a manual exception process must be initiated at the centre or awarding organisation."

5 Behaviour related to Named Orders

1. As indicated in Section 4: *Orders*, the centre will not provide learner details in a Named Order if these details have already been provided in an earlier order. The message processing component in the awarding organisation system will make use of the learner details available in its internal database.
2. If learner details are provided in a Named Order and the same learner data is already present in the awarding organisation database due to an earlier order from the same centre, then the learner data in the message may be ignored. Some awarding organisations will choose to completely ignore learner personal details when they are provided by this unapproved approach. Other awarding organisations will check the data and provide relevant feedback, but will not make any changes to the learner data within their internal database. No amendments to learner details, whether changes or inclusion of additional data, will be made by any awarding organisation unless provided using the Amend Learner Details transaction type. See Section 4: *Orders*, heading 3 *Provision of Learner Details* for further information.
3. If learner details are not provided with a Named Order because these have been provided earlier, but a required attribute is missing in the original learner data which invalidates the transaction, then a transaction level feedback is generated indicating missing learner attribute. In this case the centre will create the missing learner attribute in their system and send an Amend Learner Details transaction before sending the Named Order again.
4. If two bookings related to the same learner and the learner data contains invalid information, feedback for invalid attribute is provided against both the transactions.
5. Provision of a Confirmation message at transaction level means that the transaction has been accepted by the awarding organisation. For details on the date which will apply for charging purposes refer to Business Rule C6 in Section 4: *Orders*.

6 Centre-MIS Application Perspective

A centre is expected to receive lots of feedback messages from the awarding organisations and the MIS application should be able to process the feedback messages successfully.

The MIS application should receive all the information necessary in a feedback message from the awarding organisation in order to enable it to:

- display the feedback messages in a meaningful way to the user
- automatically suggest the user fix the data using various interfaces or functionalities available within the application (wherever possible)
- highlight any unresolved issues (wherever possible) out of the feedback received before resending the data.

The MIS application may also incorporate the following functionality:

- manage timestamp for transactional data sent to the awarding organisation
- manage timestamp for actual XML message sent to the awarding organisation
- manage timestamp for received feedback
- mark each transaction as successfully processed, or rejected, or processed with warnings
- take immediate action on transactions not processed successfully either by taking automated action or prompting the user to fix the data, or both
- interpret the feedback message. If there are any attribute level errors, the application should be able to trace back and link to the functionality so the user can correct the data or enter missing data
- verify that all the business rules which were not adhered to in the original message have been confirmed before resending the message
- alert the user to resend the corrected data within the prescribed time limit
- if a feedback is received for a missing attribute which is mandatory, alert the user to take corrective action so that the same error will not reoccur.

7 Feedback Examples

This table summarises the feedback examples described in detail below and is intended to cover a wide enough example set to illustrate the principles that should apply. This does not preclude other scenarios.

Ref	Scenario	Direction	Level/s			Severity/ies				Comments
			M	T	A	C	A	W	E	
A	Whole message rejected – transaction type not supported by AO	AO to C	✓						✓	
B	All transactions rejected – missing required data across all transactions	AO to C		✓					✓	Attribute level feedback is required to explain why the transactions are rejected, even though in this example all transactions within the message are rejected. However, as the attributes are missing, the attribute-related feedback is provided at transaction level.
C	Some transactions successfully processed, others rejected – invalid data	C to AO		✓		✓			✓	For transactions with no issues, Confirmation feedback is provided at the transaction level. For transactions with invalid data, Error feedback is provided for both the transaction and the invalid attributes. No message level feedback is provided.
					✓				✓	
D	All transactions successfully processed	AO to C	✓			✓				All transactions within the message have been processed. To avoid the need to send back the same confirmation for every transaction, a single message level confirming the whole message has been processed is sufficient. If preferred, transaction level confirmation could be sent instead.
E		AO to C		✓		✓				

Ref	Scenario	Direction	Level/s			Severity/ies				Comments
			M	T	A	C	A	W	E	
	All transactions successfully processed. For one transaction, warning given of a problem that could arise later on if no action is taken				✓			✓		Confirmation feedback is provided for all transactions. Additional warning feedback is provided to inform the centre of an issue that may cause problems later, as the attributes the warning relates to were not present in the original message.
F	Transaction successfully processed, but warning given of a subsequent problem due to a follow-on validation	AO to C		✓		✓				This message is provided at a later time and in addition to the previous feedback given in Example E. In this case the feedback is against a single transaction from the original message. An attribute that was originally accepted in Example E (a ULN) has failed further validation and therefore a warning feedback has now been provided for this attribute. This warning is accompanied by advisory feedback providing suggestions for actions to take to resolve the issue. Note that the warning is provided against an attribute in a 'related record' and so the primary key identifiers of this record are provided.
					✓		✓	✓		
G	All transactions rejected – received after processing deadline	AO to C		✓					✓	Transaction-level error feedback is provided for each transaction, indicating that the transaction has been rejected and the reason for the rejection.
H	Transaction rejected	AO to C		✓					✓	Transaction-level error feedback is provided for each transaction, indicating that the transaction has been rejected and the reason for the rejection. Feedback text includes additional explanatory information from awarding organisation system.
					✓				✓	

Ref	Scenario	Direction	Level/s			Severity/ies				Comments
			M	T	A	C	A	W	E	
I	Transaction successfully processed but some optional data provided has not been processed.	AO to C		✓		✓				Transaction level confirmation feedback supported by attribute level warning feedback that some optional data provided is invalid and has not been processed.
					✓			✓		
J	Transaction rejected as not all required information is provided	AO to C		✓					✓	Transaction-level error feedback, indicating that the transaction has been rejected and the reason for the rejection.
					✓				✓	

Table 3 Feedback Examples

7.1 Example A

Centre C1 submits Message Y comprising numerous Award Claim transactions to AO1. AO1 does not support this transaction, so the whole message is rejected.

Supported by Ref_Message_Id = X to explain which message it relates to:

0002	Message rejected	Error
0202	Prohibited transaction: Award Claim not supported by this AO	Error

7.2 Example B

Centre C1 submits Message Z comprising two Centre Assessed Outcome transactions to AO1. Both transactions relate to carry forwards, but the orders did not advise this and the new message does not contain any details of the previous centre, or candidate numbers, or other details. So both transactions, in this case the whole message, are rejected, but with transaction level feedback.

Supported by Ref_Message_Id = X to explain which message it relates to:

Supported by relevant transaction-identifying attributes for Transaction 1:

0201	Centre Assessed Outcome rejected	Error
6. 0292	7. You have advised that Learner zx3w434wres is carrying forward a previous result for ENGB3 but no details have been provided.	8. Error

Supported by relevant transaction-identifying attributes for Transaction 2:

9. 0201	10. Centre Assessed Outcome rejected	11. Error
0292	You have advised that Learner gy86h7gt7hfg is carrying forward a previous result for ENGB3 but no details have been provided.	Error

Although the carry-forward related feedback (0292) is attribute-level feedback from a business perspective, it is provided as transaction-level feedback from a structural viewpoint as there is no related QEOutcomeCarryForward record against which to provide attribute-level feedback. This feedback will be provided directly against the primary record for the transaction (QEOutcome).

Note that feedback code 0292 is related to all four attributes that it is possible to use to convey information about a carry-forward (Centre_Party_Id_CAO_Originator, AO_Candidate_Number_CAO, Carry_Forward_Series_Label and Carry_Forward_Additional_Details).

7.3 Example C

AO1 submits Message W comprising numerous results transactions for numerous QEs to Centre C1. Centre C1 is able to process most transactions but there are 45 transactions it cannot process because of an attribute level data issue.

Supported by Ref_Message_Id = W to explain which message it relates to:

Supported by relevant transaction-identifying attributes for Transaction 1:

0200	Results successfully processed	Confirmation

Supported by relevant transaction-identifying attributes for Transaction 2:

0200	Results successfully processed	Confirmation

Supported by relevant transaction-identifying attributes for Transaction 3:

0200	Results successfully processed	Confirmation

Supported by relevant transaction-identifying attributes for Transaction 4:

0200	Results successfully processed	Confirmation

And so on until…

Supported by relevant transaction-identifying attributes for Transaction 134:

0201	Results rejected	Error

Supported also by relevant attribute names:

- QE_Outcome_Value

0602	Grade could not be imported for Learner zx3w434wres qualification J567	Error

Supported by relevant transaction-identifying attributes for Transaction 135:

0201	Results rejected	Error

Supported also by relevant attribute names:

- QE_Outcome_Value

0602	Grade could not be imported for Learner 78guytufuf qualification J567	Error

Supported by relevant transaction-identifying attributes for Transaction 136:

0201	Results rejected	Error

Supported also by relevant attribute names:

- QE_Outcome_Value

0602	Grade could not be imported for Learner 7fvryy876g qualification J567	Error

Supported by relevant transaction-identifying attributes for Transaction 138:

0200	Results successfully processed	Confirmation

Supported by relevant transaction-identifying attributes for Transaction 139:

0200	Results successfully processed	Confirmation

Supported by relevant transaction-identifying attributes for Transaction 140:

0201	Results rejected	Error

Supported also by relevant attribute names:

- QE_Outcome_Value

0602	Grade could not be imported for Learner gut8676r5h7y qualification J567	Error

Supported by relevant transaction-identifying attributes for Transaction 141:

0201	Results rejected	Error

Supported also by relevant attribute names:

- QE_Outcome_Value

0602	Scaled/Weighted Mark could not be imported for Learner gut8676r5h7y qualification F50301	Error
0606	The value Scaled/Weighted Mark is invalid; it exceeds the permitted maximum for Scaled/Weighted Mark	Error

Supported by relevant transaction-identifying attributes for Transaction 142:

0201	Results rejected	Error

Supported also by relevant attribute names:

- QE_Outcome_Value

0602	Scaled/Weighted Mark could not be imported for Learner gi7ri6tg78 qualification F50301	Error
0606	The value Scaled/Weighted Mark is invalid; it exceeds the permitted maximum for Scaled/Weighted Mark	Error

And so on, for each remaining transaction, providing either transaction level confirmation, in which case no further feedback is needed for that transaction, or transaction level rejection, in which case multiple feedback at transaction and/or attribute level will be provided as necessary to highlight each issue.

7.4 Example D

Centre C1 submits Message U comprising numerous Award Claim transactions to AO2. AO2 supports this transaction and all the data passes validation with no issues. Because all transactions in the message were processed successfully, and there is no further feedback

needed, a message level confirmation will suffice (though the awarding organisation could instead provide transaction level confirmation against every transaction).

Supported by Ref_Message_Id = U to explain which message it relates to:

0001	Message successfully processed	Confirmation

7.5 Example E

Centre C1 submits Message V comprising four Named Order transactions to AO3. These are for four completely new learners so learner details are included. AO3 supports Named Order entries for the relevant QEs and almost all the data passes validation with no issues. In one case the centre has indicated a carry forward applies but has not provided the previous details for the carry forward (centre party identifier, AO candidate number, series label or additional details): this does not prevent the Named Order being processed but a warning is returned to the centre because if this data is not provided with the CAO in due course, there will be a hold up to processing the learner's result.

Supported by Ref_Message_Id = U to explain which message it relates to:

Supported by relevant transaction-identifying attributes for Transaction 1:

0203	Named Order (Entry) successfully processed	Confirmation
0292	You have advised that Learner YTV865T975 is carrying forward a previous result for HIST01 but no details have been provided.	Warning

Although the carry-forward related feedback (0292) is attribute-level feedback from a business perspective, it is provided as transaction-level feedback from a structural viewpoint as there is no related QEOutcomeCarryForward record against which to provide attribute-level feedback. This feedback will be provided directly against the primary record for the transaction (QELearnerBooking).

Note that feedback code 0292 is related to all four attributes that it is possible to use to convey information about a carry-forward (Centre_Party_Id_CAO_Originator, AO_Candidate_Number_CAO, Carry_Forward_Series_Label and Carry_Forward_Additional_Details).

Supported by relevant transaction-identifying attributes for Transaction 2:

0203	Named Order (Entry) successfully processed	Confirmation

Supported by relevant transaction-identifying attributes for Transaction 3:

0203	Named Order (Entry) successfully processed	Confirmation

Supported by relevant transaction-identifying attributes for Transaction 4:

0203	Named Order (Entry) successfully processed	Confirmation

7.6 Example F

This follows on from Example E above. Centre C1 previously submitted Message V comprising four Named Order transactions to AO3. Now that follow-on validation has occurred between AO3 and the LRS, it turns out that the ULN provided in transaction 3 cannot be successfully

matched. In practice it may be that the awarding organisation decides that, as they have already processed the order and as the QE does not mandate a ULN to be provided, they will just phone Centre C1 to resolve the issue but ideally they would send relevant feedback as below:

Supported by Ref_Message_Id = U to explain which message it relates to:

Supported by relevant transaction-identifying attributes for Transaction 3:

0203	Named Order (Entry) successfully processed	Confirmation

Supported also by primary key attribute names/values for 'related' PartyRelationshipRole record:

- Party_Id_1st ('LRS')
- Party_Id_2nd (same as QELearnerBooking Learner_Party_Id)
- Party_Role_Type ('Learner')

Supported also by relevant attribute names/values:

- Relationship_Reference (value of ULN provided in original message)

0133	ULN failed validation with LRS	Warning

This may be supplemented also by:

0701	Please contact the Learner Registration Service (LRS) for guidance	Advisory
0702	Please submit an amendment to resolve this issue	Advisory

7.7 Example G

Centre C1 submits message T comprising various Centre Assessed Outcome transactions to AO4. But these are provided after the internal order processing deadline for the awarding organisation, the end of which reflects the deadline or barring date.

Supported by Ref_Message_Id = T to explain which message it relates to:

Supported by relevant transaction-identifying attributes for Transaction 1:

0201	Centre Assessed Outcome rejected	Error
0555	Barring date has passed for this series/availability period	Error

Supported by relevant transaction-identifying attributes for Transaction 2:

0201	Centre Assessed Outcome rejected	Error
0555	Barring date has passed for this series/availability period	Error

Supported by relevant transaction-identifying attributes for Transaction 3:

0201	Centre Assessed Outcome rejected	Error
0555	Barring date has passed for this series/availability period	Error

And so on for each transaction.

7.8 Example H

Centre C1 submits a single Named Order for a QE (QE1234) that is not permitted because the learner has an existing entry for another QE (QE6789) and the combination of these two QEs is prohibited in the same series.

Supported by Ref_Message_Id = T to explain which message it relates to:

Supported by relevant transaction-identifying attributes for the referenced transaction:

0204	Named Order (Entry) rejected	Error
0285	QE1234 is prohibited in the same series as QE6789	Error

At the moment the second (existing) QE in the prohibited pair is provided in readable feedback text. In the future this information could also be provided in machine-readable format using an 'FB_AdditionalInformation' block.

7.9 Example I

Centre C1 submits a single Named Order for a learner. As it is the first A2C transaction for this learner, the centre provides all available learner details, including the learner's UCI. Both the UCI and the learner's email address fail validation as they are not the correct format, but the transaction is accepted, as neither is required for the QE for which the entry is being made.

Supported by Ref_Message_Id = T to explain which message it relates to:

Supported by relevant transaction-identifying attributes for Transaction:

0203	Named Order (Entry) successfully processed	Confirmation

Supported by record-identifying attributes for 'related' PartyRelationshipRole record having Party_RR_Reference_Type of 'UCI':

- Party_Id_1st ('JCQ');
- Party_Id_2nd (same as QELearnerBooking Learner_Party_Id);
- Party_Role_Type ('Learner').

Supported by relevant attribute names/values:

RelationshipReference ('UC789123')

0128	UCI supplied for learner rejected	Warning
0127	Supplied UCI of UC789123 is not in valid format.	Advisory

Supported by record-identifying attributes for 'related' locator record:

- Locator_Id.

Supported by relevant attribute names/values:

- Locator_Type ('Email Address')

- Email_Address ('joe.bloggs').

0106	email address not in valid format	Warning

7.10 Example J

This example follows on from Example I. Before receiving the feedback for transaction L, the centre submits another Named Order for the same learner; this time for a QE which does require the learner's UCI to be provided. As the centre sent the UCI for the learner in the previous transaction, it does not resend it in this transaction. As the awarding organisation does not have a valid UCI for the learner and the QE requires one, the transaction is rejected.

Supported by Ref_Message_Id = T to explain which message it relates to:

Supported by relevant transaction-identifying attributes for Transaction:

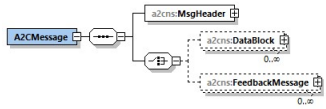
0204	Named Order (Entry) rejected	Error
0164	UCI missing	Error

Although the UCI-related feedback is attribute-level feedback from a business perspective, it is provided as transaction-level feedback from a structural viewpoint as there is no related PartyRelationshipRole record against which to provide attribute-level feedback. The following specific feedback will be provided directly against the primary record for the transaction (QELearnerBooking).

Note that feedback code 0164 is a template message - the receiving system must interrogate the feedback message to determine that the missing Party_RR_Reference_Type is a UCI.

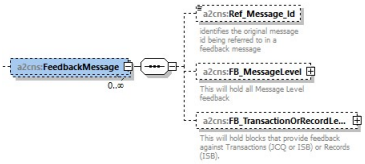
8 Description of Feedback Schema

8.1 Feedback within the overall Message Structure



A message comprises a 'MsgHeader' block, a set of 'DataBlock' blocks (for most messages), and a 'FeedbackMessage' block (for feedback messages). All feedback related information is provided within the 'FeedbackMessage' block.

8.2 The FeedbackMessage Block



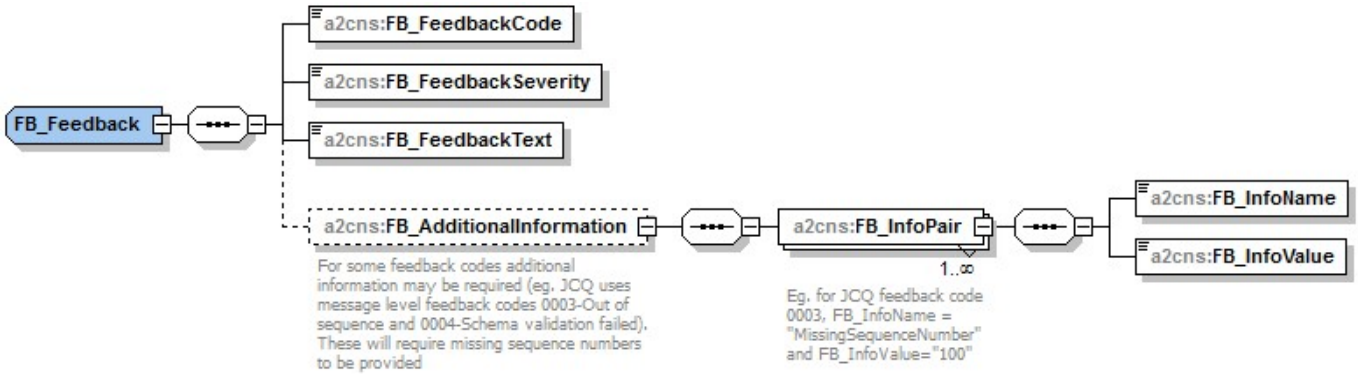
All feedback within a feedback message will be contained within the 'FeedbackMessage' block. Within this top-level container, all message-level feedback will be grouped under a single 'FB_MessageLevel' block and all transaction-level or record-level feedback under a single 'FB_TransactionOrRecordLevel' block.

Note: For A2C this block will contain feedback related to a transaction only. For ISB this block will contain feedback against a record within a data block. (The details of the ISB feedbacks will be published by ISB separately. Reference to record level feedback is given for information only in this document.)

8.3 Message-level Feedback

All message-level feedbacks will be contained within a single 'FB_MessageLevel' block, with each feedback contained in its own 'FB_Feedback' block, each corresponding to a single feedback code. All feedback codes relate directly to the message and will never be associated with an attribute. Separate feedback codes have been defined for each message header attribute for which feedback may be provided. In the above diagram, the details of the 'FB_Feedback' structure have been collapsed (yellow box). This structure is expanded below. Note that this 'FB_Feedback' structure will be reused for all levels of feedback.

8.4 Common Feedback Structure for all Levels of Feedback



The 'FB_Feedback' data structure has been defined as a reusable XML data type, and this same data structure will be reused when providing message, transaction, record or attribute-level feedback. Information about the feedback level is not included in the 'FB_Feedback' structure, as this will be known from the block it appears under (eg 'FB_MessageLevel' for message-level feedback).

An 'FB_Feedback' block contains

i. a mandatory feedback code

ii. a mandatory feedback severity

iii. a mandatory feedback text.

iv. It may also contain an optional 'FB_AdditionalInformation' block, which holds one or more name/value pairs providing additional information relevant to the feedback. For example, the out of sequence and schema validation failure messages will use this block to provide information about the sequence number of the last successfully processed message, as well as the sequence numbers of missing and pending messages (Refer to Section 11: *Solutions Architecture* for more information).

For each 'FB_InfoPair' pair, the 'FB_InfoName' element will hold a commonly agreed identifier for the information (eg 'LastProcessedSequenceNumber') and the 'FB_InfoValue' element will contain the corresponding value (eg '100'). For feedback codes 0003 and 0004, the following identifiers are used:

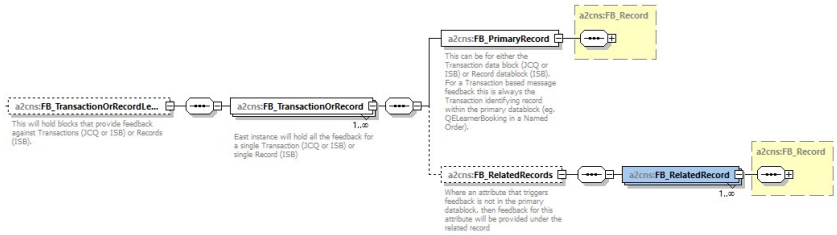
- LastProcessedSequenceNumber – For the last successfully processed sequence number
- MissingSequenceNumber – For missing sequence number
- PendingSequenceNumber – For sequence number of the messages pending to be processed.

In some cases (eg missing and pending sequence numbers) multiple values of the same type may need to be provided. In these cases, there will be multiple 'FB_InfoPair's with the same 'FB_InfoName' (eg 'MissingSequenceNumber') but different 'FB_InfoValue' data (eg '101', '102', '103').

The 'FB_AdditionalInformation' block is used:

- to provide some of the extra information included in readable feedback text for template feedback codes.
- where more information could be supplied to help the users of the original sending system make informed decisions.

8.5 Transaction-related Feedback



For A2C, all feedback codes that are not message level must be related to a transaction. All feedback codes related to a single transaction will be grouped together under a single 'FB_TransactionOrRecord' block. A transaction has a primary transaction defining record in the primary data block (eg a QELearnerBooking record for Named Order transactions). However each transaction can also have several 'related records' (eg QEBooking or PartyRelationshipRole records for Named Order transactions). Therefore each 'FB_TransactionOrRecord' block will contain a mandatory 'FB_PrimaryRecord' block and an optional 'FB_RelatedRecords' block containing feedback associated with one or more related records[2]. It has been agreed that ISB record-level feedback will use exactly the same data structure, hence the use of the composite 'TransactionOrRecord' naming convention. Note that a common data structure has been chosen to contain the details of feedback provided under the primary transaction identifying record and feedback provided under 'related records' for a transaction. This common 'FB_Record' structure has been collapsed (yellow boxes), but is expanded below.

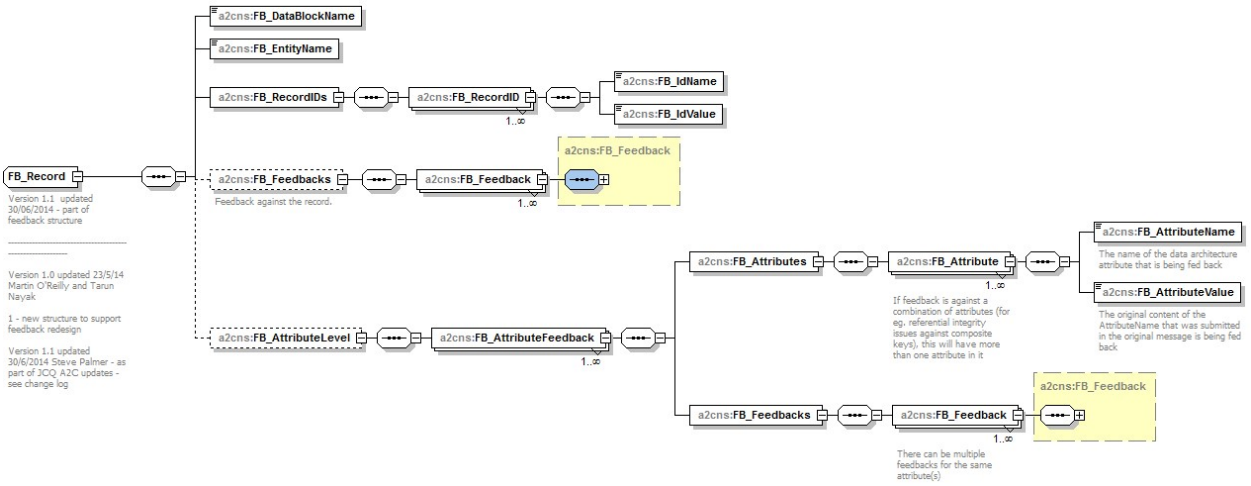
8.6 Common Structure for Transaction Feedback on Primary and Related Records

Feedback related to a transaction can either be:

1. Feedback directly related to the transaction itself (eg <Transaction_Type> rejected). This requires information to identify the primary transaction identifying record.
2. Feedback related to an attribute within the primary transaction-defining record (eg 'Learner_Assmnt_Start_Date_Time' is within the primary QELearnerBooking record for a Named Order). This requires information to identify both the primary transaction identifying record and the attribute within that record.
3. Feedback related to an attribute within a 'related record' for the transaction (eg 'QE_Booking_Type' is within the related QEBooking record for a Named Order). This requires information to identify both the related record and the attribute within that record.

Due to the similarities in the information required in all three cases, a single data structure has been designed to handle all three cases. This data structure has been defined as a reusable XML data type named 'FB_Record'. This data structure is illustrated below.

[2] Note that attributes missing from a related record are a special case. See the specific guidance on 'Handling Missing Data' in the 'Feedback Behaviour' section of this document (Heading 4 Point 25).



For A2C, each block of type 'FB_Record' will contain all the feedback associated with either

(i) the primary record for a transaction

or

(ii) a 'related record' for a transaction

The 'FB_Record' data structure contains the following elements:

a FB_DataBlockName
This contains the name of the data block.

b FB_EntityName
This contains the name of the entity within the data block which contains the record

c FB_RecordID
This contains the names and values of the primary key attributes for the record.

d FB_Feedback
When the 'FB_Record' structure is used for the primary transaction-identifying record for a transaction, this contains one or more transaction-level feedbacks directly related to the transaction (case 1 above). Each feedback is represented by the same 'FB_Feedback' data structure used for message-level feedback. For A2C, when the 'FB_Record' structure used for a 'related record' for a transaction the 'FB_Feedbacks' block will not be used, as there is no record-level feedback for JA2C.

e FB_AttributeLevel
This is an (optional) container that holds one or more attribute level feedbacks. Each attribute-level feedback consists of one or more name/value pairs for the attribute(s) that triggered the feedback. For most feedbacks, there will only be a single feedback-triggering attribute. However, referential integrity issues are related to groups of 'foreign key' attributes, and therefore the associated feedback must include multiple attribute name/value pairs. Each attribute (or set of attributes) can trigger multiple feedbacks and therefore each 'FB_AttributeFeedback' contains an 'FB_Feedbacks' block that can hold one or more 'FB_Feedback' elements. Again, the same 'FB_Feedback' structure is used for attribute-level feedback as has been used for message, transaction and record-level feedback.

Due to the reuse of the 'FB_Record' data structure for both the transaction-identifying 'FB_PrimaryRecord' and the related 'FB_RelatedRecord'(s), attribute-level feedback can be provided for both primary and related data blocks within a transaction, covering cases 2 and 3 above.