E-Appeal
Electronic Record on Appeal
Phase 1 – Record On Appeal Transmission Specification
An Information Exchange Package (IEP)

Submitted to the Technical Advisory Council for review on August 4, 2006

This document defines requirements for transmitting a Record on Appeal (ROA) composed of electronic documents using IBM MQSeries as the transport infrastructure within the Arizona Judicial Information Network (AJIN).

The E-ROA XML (defined below) will be used to 'contain' the ROA metadata, the Index of Record (IOR) document, and all documents and document metadata comprising the record, or an alteration of the record.

Since there is no limit on the number or size of documents which may need to be transmitted, this specification considers that multiple MQ message groups may be required to fully transmit the ROA.

An MQ message group will always be used even when there is a single message and no segmentation is required or used.

The first MQ message in each group will contain the E-ROA XML. For the first MQ message group, the second message in the group will contain the Index of Record document content. The following messages will each contain a single document unless that document is too large for a single message or the document is too large for the current message group. When a document is too large for a single MQ message, but not too large for the message group, the submitting application will segment the document into multiple MQ messages, utilizing the fewest MQ messages necessary to contain the entire document. When the document is too large for the current message group, then a new Submission/MQ message group will be initiated. If a document is too large to be transmitted by itself within a single MQ message group, then the document must be transmitted using other means (such as DVD-ROM, etc.)

MQ will be permitted to apply arbitrary segmentation to facilitate efficient message transport.

When documents are put to the queue, a commit will be issued after the successful put of each document. If a document has been segmented by the putting application, the commit will follow the successful put of the last document segment for the document.

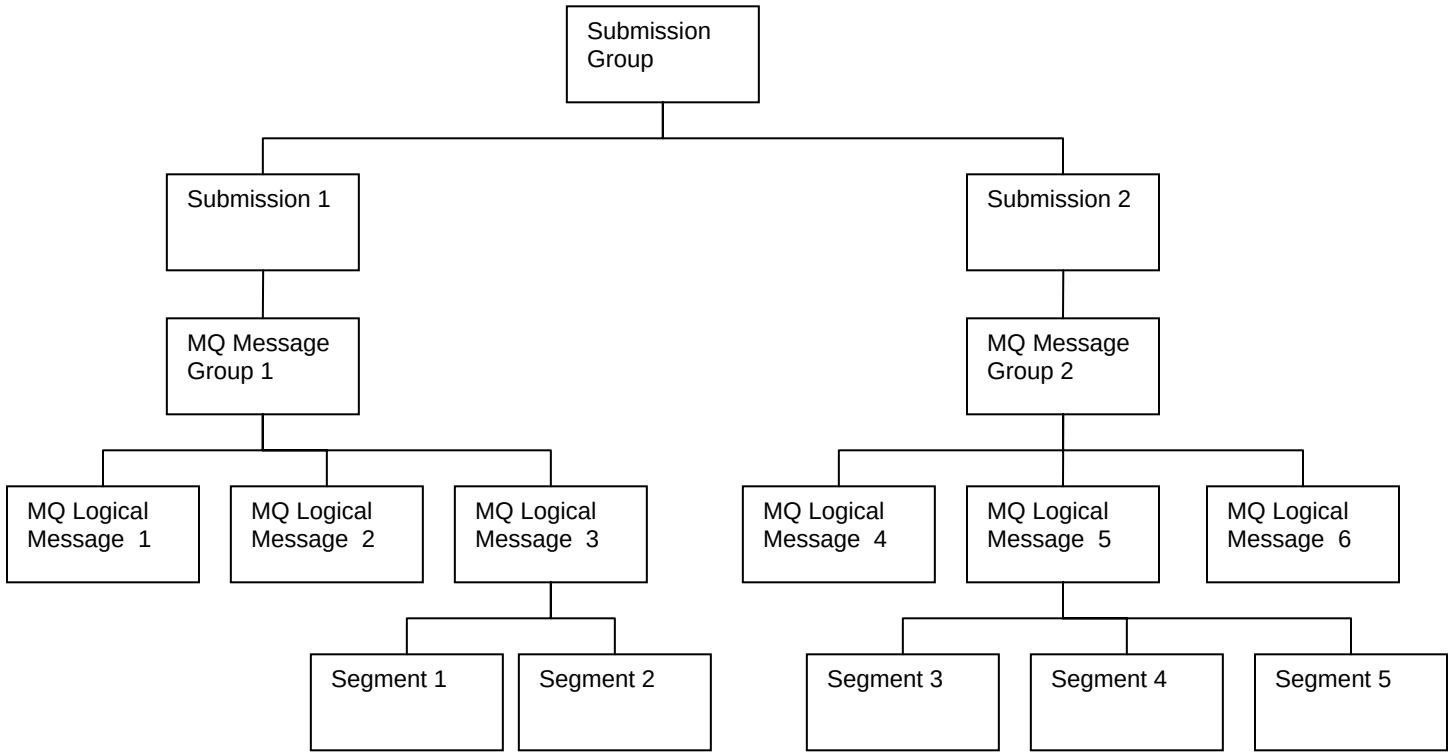
When messages are read from the queue, the message will not be permanently removed from the queue until all messages within the MQ Message group have been read, and successfully ‘fielded’.

All MQ messages will be persistent.

MQ Message Identity Context must refer to the original sending application (such as the ‘Assembler’). If the message originating application uses an agent application (such as an MQputter) to ‘put’ the message to MQ, then the Identity Context should not refer to the putting agent, but should instead identify the message originator.

This specification complies with GJXDM naming conventions. Whenever possible, GJXDM element names were used.

The following chart illustrates the relationship between MQ and the E-ROA XML structure.



A Submission Group is comprised of all the Submissions necessary to transmit the ROA transaction. Each Submission is handled by an MQ message group. The Message group contains logical MQ messages which may be broken into segments. Generally there will be one document (i.e. XML document, Index of Record document, Record on Appeal document, or Supplemental Item document) per logical message. MQ Logical Message 1 will contain the E-ROA XML document compliant with this specification. MQ Logical Message 4 will also contain an E-ROA XML document per this specification. MQ Logical Message 2 will contain the Index of Record document. There will only be one Index of Record document for the whole Submission group.

E-ROA XML Document Type Definition (DTD)

The following Document Type Definition (DTD) expresses the transmission XML requirements defined further within this specification:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!--copyright @ 2005, Arizona Supreme Court. All Rights Reserved. -->

<!ELEMENT Submission (SubmissionInfo, IndexOfRecord, SupplementalItems?)>
     <!ATTLIST Submission     type CDATA #FIXED 'ROA'
                              version CDATA #FIXED '1.0'
                              status CDATA #REQUIRED
                              xml:base CDATA #IMPLIED >
   <!ELEMENT SubmissionInfo (SubmissionCourtName, SubmissionDateTime, SubmissionPersonName,
SubmissionSeqNum)>
      <!ELEMENT SubmissionCourtName (#PCDATA)>
         <!ATTLIST SubmissionCourtName code CDATA #REQUIRED>
      <!ELEMENT SubmissionDateTime (#PCDATA)>
          <!—Use ISO 8601 format, i.e. ccyy-mm-ddThh:mm:ss[.ttt] -->
      <!ELEMENT SubmissionPersonName (#PCDATA)>
      <!ELEMENT SubmissionSeqNum (#PCDATA)>
   <!ELEMENT IndexOfRecord (CaseNumber, CaseTitleText, AppellateCourtName, AppellateCaseNumber?,
                            DocumentContent?, Transaction+)>
      <!ATTLIST IndexOfRecord id ID #REQUIRED
                              version CDATA #REQUIRED
                              groupCount CDATA #REQUIRED
                              docCount CDATA #REQUIRED >
      <!ELEMENT CaseNumber (#PCDATA)>
        <!ATTLIST CaseNumber displayMask CDATA #IMPLIED >
      <!ELEMENT CaseTitleText (#PCDATA)>
      <!ELEMENT AppellateCourtName (#PCDATA)>
        <!ATTLIST AppellateCourtName code CDATA #IMPLIED >
      <!ELEMENT AppellateCaseNumber (#PCDATA)>
      <!ELEMENT Transaction (IndexOfRecordDocument+)>
        <!ATTLIST Transaction type CDATA #FIXED 'ROA'
                              id ID #REQUIRED
                              directive (FILE|AMEND-ADD|AMEND-REPLACE|AMEND-REMOVE|AMEND-INSERT) #REQUIRED
                              content  (COMPLETE|SEGMENTED) #REQUIRED >
          <!ELEMENT IndexOfRecordDocument (DocumentIndexNumber, Document)>
            <!ELEMENT DocumentIndexNumber (#PCDATA)>
```

```
          <!ELEMENT Document (DocumentTypeText, DocumentSubtypeText?, DocumentTitleText,
                              DocumentFiledDate, DocumentAbstractText?, DocumentCommentText?,
                              DocumentSealedIndicator?, DocumentPartition+)>
            <!ATTLIST Document id ID #REQUIRED>
              <!ELEMENT DocumentTypeText (#PCDATA)>
              <!ELEMENT DocumentSubtypeText (#PCDATA)>
              <!ELEMENT DocumentTitleText (#PCDATA)>
              <!ELEMENT DocumentFiledDate (#PCDATA)>
                     <!—Use ISO 8601 format, i.e. ccyy-mm-dd -->
              <!ELEMENT DocumentAbstractText (#PCDATA)>
              <!ELEMENT DocumentSealedIndicator (#PCDATA) >
              <!ELEMENT DocumentPartition (DocumentFileName, DigestMethod?, DigestValue?,
                                  DocumentPartitionNumber?, DocumentContent)>
                 <!ELEMENT DocumentFileName (#PCDATA)>
                 <!ELEMENT DigestMethod (#PCDATA)>
                   <!ATTLIST DigestMethod algorithm CDATA #REQUIRED>
                 <!ELEMENT DigestValue (base64)>
                 <!ELEMENT DocumentPartitionNumber (#PCDATA)>
                 <!ELEMENT DocumentContent EMPTY>
                   <!ATTLIST DocumentContent id ID #REQUIRED
                                             mimeType CDATA #REQUIRED
                                             size CDATA #IMPLIED
                                             xmlns:xlink CDATA #FIXED ‘http://www.w3.org/1999/xlink’
                                             xlink:type CDATA #FIXED ‘simple’
                                             xlink:href CDATA #REQUIRED >
<!ELEMENT SupplementalItems (DocumentCommentText?, Document+)>
   <!ATTLIST SupplementalItems   docCount CDATA #REQUIRED >
      <!ELEMENT DocumentCommentText (#PCDATA)>
```

Elements

Submission is the root element of the E-Record on Appeal XML for the purpose of transmitting the record on appeal from a trial court to an appellate court or from one appellate court to a higher appellate court. In general, a Submission is a single tendering of an electronic filing to a court. In the context of a Record on Appeal filing, as related to this specification, a Submission is a single or partial Record on Appeal transmission. When all of the documents which comprise the Record On Appeal transaction are able to be transmitted in a single communication, then a Submission will hold the entire transaction. When this is not possible or practical due to limitations of the supporting hardware and software infrastructure, then a Submission may only contain a partial Record on Appeal transaction.

Although the Submission may be received by the destination court, it may not be accepted for filing at that court. All Submissions are subject to inspection and review by the receiving court prior to determining acceptance or rejection. Upon acceptance, a Submission should be 'filed' at the appellate court. If not accepted, the Submission should be rejected. Submission review, followed by either acceptance or rejection, should be conducted in a timely manner.

A Submission may carry a complete Record On Appeal for filing in an appellate court. A Submission may also carry a partial Record on Appeal which may supplement the record, or alter the record. Alteration of the record may include adding documents, removing documents, or substituting documents.

A complete Record On Appeal from a trial court may contain many documents. Currently there are no limits on the number or the size of the documents contained in the record. As such, a complete Record On Appeal may exceed the transport capacity of the communication infrastructure (i.e. such as MQSeries). When not practical to transmit an entire Record on Appeal due to size and capacity limitations, the record, or an alteration to the record, may be transmitted in more than one Submission. When more than one Submission is used to transmit the record or a record alteration, then the element values for SubmissionCourtName, and CaseNumber, and the attribute value of the IndexOfRecord id attribute of all Submissions must match (i.e. contain equal values), and the SubmissionSeqNum should be used to properly sequence the Submissions. These related Submissions are referred to as a 'group of Submissions' (or just a 'Submission group').

The Submission element has three (3) attributes:

type identifies the kind of Submission as 'ROA'. If this attribute is present, it must have a value of 'ROA'. If absent, is assumed to be 'ROA'.

version identifies the version of the XML data definition as version 1.0. If this attribute is present, it must have a value of "1.0". If absent, is assumed to be "1.0".

status identifies the current state of the Submission. When the Submission XML is first created, the status should be "CREATED". Immediately prior to tendering the Submission to the

Transport (i.e. MQSeries), the status should be set to “SENT”. If an application sends a Submission immediately following the Submission creation, then the “CREATED” status need not be set. Where possible, upon successful delivery of the Submission to its destination, the Transport should set the status to “DELIVERED”. When the submission is retrieved from the Transport, the reader or fielder application should set the status to “RECEIVED”. Subsequent processing applications may define additional status values. Upon completion of final processing of the Submission, the status should be set to “COMPLETED”, unless the XML document is immediately destroyed.

xml:base identifies a base URL that will be applied to all relative XLINK URLs. The xml:base attribute is not applied to namespace URLs, or any URLs outside the root element. Note that the xml namespace need not be declared and is pre-bound to the URI http://www.w3/org/XML/1998/namespace.

```
<!ELEMENT Submission (SubmissionInfo, IndexOfRecord, SupplementalItems?)>
    <!ATTLIST Submission     type CDATA #FIXED “ROA”
                             version CDATA #FIXED “1.0”
                             status CDATA #REQUIRED
                             xml:base CDATA #IMPLIED >
```

SubmissionInfo holds additional elements related to the Submission.

```
<!ELEMENT SubmissionInfo (SubmissionCourtName, SubmissionDateTime, SubmissionPersonName,
SubmissionSeqNum)>
```

SubmissionCourtName identifies the court submitting the record on appeal transaction. Often the submitting court will be a trial court, but the SubmissionCourtName could be an appellate court, such as when a Petition for Review has been filed at the Supreme Court emanating from a case on appeal at the court of appeals. This element should be populated with the name of the court, such as “Mohave Superior Court”.

The SubmissionCourtName element has a single attribute:

code identifies the submitting court using the standard ‘Court Code’ value as defined by the Arizona State Judicial system’s standard appellate case management system Appellamation. Specifically, the code attribute value must be a valid value from the ‘court_cd’ column of the ‘psi.acm_court’ table within the Appellamation ‘appellaproduction’ database. A valid value will not have its ‘disabled_usage_flg’ column value set to ‘Y’.

```
<!ELEMENT SubmissionCourtName (#PCDATA)>
```

```
<!ATTLIST SubmissionCourtName code CDATA #REQUIRED>
```

SubmissionDateTime indicates the date and time that the Submission document (i.e. “E-ROA”, the XML conforming to this specification) was created. Must use ISO 8601 format. Must provide century, year, month, days, hour, minute and seconds components. The fractional seconds part is optional.

```
<!ELEMENT SubmissionDateTime (#PCDATA)>
    <!—Use ISO 8601 format, i.e. ccyy-mm-ddThh:mm:ss[.ttt] -->
```

SubmissionPersonName identifies the person that created the Submission document.

```
<!ELEMENT SubmissionPersonName (#PCDATA)>
```

SubmissionSeqNum identifies the order in which the current Submission fits into a group of Submissions (i.e. a Submission group), when it is necessary to use multiple Submissions to send a Record on Appeal or a record alteration. If a single Submission is used to transmit the record or an alteration of the record, then SubmissionSeqNum should be “1”, otherwise SubmissionSeqNum should be an integer value which is one unit greater than the value of the SubmissionSeqNum of the Submission that immediately preceded it, with the first Submission of the group having a SubmissionSeqNum value of “1”.

```
<!ELEMENT SubmissionSeqNum (#PCDATA)>
```

IndexOfRecord identifies a structure which holds the Index of Record ‘document content’ and all the documents contained within the record which are included in the Submission. The IndexOfRecord specifies the SubmissionCourtName case, by case number and title, and holds the Index of Record ‘document content’. The contents and format of the Index of Record document are largely set by Supreme Court Administrative Order 99-75. DocumentContent for the IndexOfRecord must be provided if the SubmissionSeqNum is “1”, otherwise the IndexOfRecord DocumentContent element should be empty. If the IndexOfRecord contains multiple Transactions, then the IndexOfRecord DocumentContent should reflect the state of the record upon successful acceptance of all the Transactions contained within the IndexOfRecord (or all the IndexOfRecord elements within the Submission group).

The IndexOfRecord contains four (4) attributes:

id uniquely identifies an IndexOfRecord. This attribute, in conjunction with SubmissionCourtName and CaseNumber, is used to identify Submissions which comprise an application segmented Record on Appeal filing (i.e. a Submission group). As such, the

IndexOfRecord ID attribute value must be unique for the combination of SubmissionCourtName and CaseNumber, and constant across all Submissions within the group. This further constrains the values permitted for the ID attribute beyond those imposed by the XML 1.0 specification which requires that: "Values of type IDR must match the Name production. A name must not appear more than once in an XML document as a value of this type; i.e. ID values must uniquely identify elements which bear them.[1]"

To further clarify this uniqueness requirement, consider that multiple Submissions from a given SubmissionCourtName for a given CaseNumber may be transmitted. This can arise from multiple scenarios, such as:

1. An initial Submission files a Record on Appeal, then later, a subsequent Submission alters the record (e.g. supplements the record).

2. A given SubmissionCourtName case becomes the lineage predecessor of multiple appellate cases. When the SubmissionCourtName case is first appealed, a Submission will follow to file the Record on Appeal. When the second appellate case is filed, a Submission may also follow filing the Record on Appeal in the second appellate case.

In each of these scenarios, the SubmissionCourtName and CaseNumber will be the same for both Submissions, but the IndexOfRecord id attribute must have a unique value for each of the separate submissions. If a unique value is not provided, then it may not be possible for the receiving application to be able to distinguish the group to which a Submission belongs. Note that 'group' as used here refers to a Submission group and should not be confused with an MQ message group. An MQ message group, when used with this specification, will identify all the MQ messages that comprise a single Submission.

version identifies the rendition of the Index of Record DocumentContent. With an initial filing of the Index of Record, it is expected that version would be "1". For additional amendment Submissions, the version should be numerically increased. Only whole integer values should be used.

groupCount identifies the total number of Submissions in the Submission group. When a Record on Appeal is segmented across multiple Submissions, groupCount identifies the number of Submissions within the group that must be received to comprise the complete Record on Appeal (or record alteration). The SubmissionSeqNum of the last Submission within a Submission group should be equivalent to groupCount.

[1] See "Extensible Markup Language (XML) 1.0, W3C Recommendation 10-February-1998", Production [56].

docCount identifies the total number of IndexOfRecordDocuments included in the Submission group. When a single Submission is used to FILE the Record on Appeal, docCount should be the number of IndexOfRecordDocument elements, and should equal the largest DocumentIndexNumber value.

```
<!ELEMENT IndexOfRecord (CaseNumber, CaseTitleText, AppellateCourtName, AppellateCaseNumber?,
                         DocumentContent?, Transaction+)>
  <!ATTLIST IndexOfRecord id ID #REQUIRED
                          version CDATA #REQUIRED
                          groupCount CDATA #REQUIRED
                          docCount CDATA #REQUIRED >
```

CaseNumber specifies the designator number used by the trial court to identify the case. The value of the element should be its basic representation without the presence of readability characters. For example, the value should be “CR0200400100”, and not “CR-200400100”, or “CR-0200400100”, or “S-1400-CR-0200400100”. CaseNumber has a single attribute:

displayMask identifies the presentation that should be used when displaying the CaseNumber. The displayMask should use text patterns to define the presentation. The text pattern should consist of *metacharacters* which have special meaning.

These metacharacters are:

@	any character
#	a number (0 – 9)
all other characters represent themselves	

displayMask example:

Using mask: @@-#####-##### with: CR0200400100

Produces: CR-02004-00100

```
<!ELEMENT CaseNumber (#PCDATA)>
  <!ATTLIST CaseNumber displayMask CDATA #IMPLIED >
```

CaseTitleText is a short name for the SubmissionCourtName case identified by CaseNumber.

```
<!ELEMENT CaseTitleText (#PCDATA)>
```

AppellateCourtName identifies the appellate court to which the Submission is directed.

The AppellateCourtName element has a single attribute:

code identifies the receiving appellate court using the standard ‘Court Code’ value as defined by the Arizona State Judicial system’s standard appellate case management system Appellamation. Specifically, the code attribute value must be a valid value from the ‘court_cd’ column of the ‘psi.acm_court’ table within the Appellamation ‘appellaproduction’ database. A valid value will not have its ‘disabled_usage_flg’ column value set to ‘Y’. The code value must represent an appellate court (must have a value between 100 and 199 inclusive, for the ‘court_level_value’ column in ‘psi.acm_court’).

```
<!ELEMENT AppellateCourtName (#PCDATA)>
  <!ATTLIST AppellateCourtName code CDATA #IMPLIED >
```

AppellateCaseNumber identifies the appellate case to which the Submission is to be directed. In the circumstance of an initial Record on Appeal filing, this would be the appellate case created as a result of a notice of appeal emanating from the SubmissionCourtName case. Typically, the appellate case will have a lineage relationship with the SubmissionCourtName case, which will generally be the appellate cases’ predecessor. The AppellateCaseNumber element is optional. When not used, the receiving court has discretion to apply the transaction to any case at the appellate court which has a lineage relationship with the SubmissionCourtName case, or to establish a new lineage relationship to an existing case at the receiving appellate court, or to create a new appellate case with a lineage relationship. Use of this element is therefore recommended. If used, then the value must be the basic representation of the case number without readability characters or other qualifiers (e.g. subtype code). For example, the case number should be “CR040123” and not “CR-04-0123”, or “CR 04-123 PRPC”, or “1-CA-CR 04-0123 PRPC”.

Even though AppellateCaseNumber is optional, if the Submission is part of a Submission group, then all Submissions within the group must have the same AppellateCaseNumber value. Therefore, if AppellateCaseNumber is populated for one of the Submissions within a group, then it must be populated for all Submissions within the group (and must be the same value).

```
<!ELEMENT AppellateCaseNumber (#PCDATA)>
```

Transaction is used to specify the type of processing that should occur upon acceptance of a Submission or a group of Submissions. The Transaction element is nested within the IndexOfRecord element and contains the IndexOfRecordDocument element. For an IndexOfRecord, at least one Transaction must be defined, but more than one Transaction is also valid (depending upon the type of Transaction and/or the ordering of Transactions). If multiple Transactions are provided, they shall be processed in the order presented, and shall represent a single logical unit of work, being either committed or rolled back as a unit. Nesting Transactions within IndexOfRecord permits multiple Transactions to apply to a single IndexOfRecord DocumentContent which represents the successfully completed state of the Record on Appeal, following the application of all the Transactions. A single Document (identified by id) must not be transacted more than one time in a Submission or group of Submissions (e.g. a Document cannot be added, then subsequently removed). The Transaction element has four (4) attributes:

type identifies the nature of the Transaction. If this attribute is present, it must have a value of 'ROA'. If absent, is assumed to be 'ROA'.

id uniquely identifies the Transaction. When a Transaction is SEGMENTED across multiple Submissions within a group, the id attribute is used to identify the Transaction which began in a prior Submission, but not yet completed. Whenever possible, Transactions should be COMPLETE and not SEGMENTED.

directive identifies what is to be done by the Transaction. An enumerated list of values exists for directive, and only values from this list may be used:

- FILE — this Submission is attempting to file a complete Record on Appeal. Due to size and capacity limitations the entire contents of the Record on Appeal may not be able to be held by a single Submission. When this occurs, then a group of Submissions will be used to fulfill the FILE Transaction directive. When the FILE directive is used, then multiple Transaction elements within the IndexOfRecord are prohibited, and the Transaction directive for all Submissions in the group must be FILE.

- AMEND-ADD — this Submission is altering an existing Record on Appeal filing by appending additional Documents to the end of the listing. The DocumentIndexNumber values for the additional Documents provided must be logically consecutive to the DocumentIndexNumber of the previous last Document in the Index of Record. This consecutive DocumentIndexNumber requirement precludes the use of a Transaction directive of AMEND-INSERT or AMEND-REMOVE prior to an AMEND-ADD since these other directives will affect index numbering. The AMEND-ADD Transaction directive must not be used with any other AMEND-ADD Transaction directives for an IndexOfRecord, either within a single Submission or within a Submission group.

AMEND-REPLACE this Submission is altering an existing Record on Appeal filing by replacing a existing document within the record, by a new Document or a revision of an existing Document. The replaced Document retains the Index of Record index number of the replaced Document, however, the DocumentTitleText, DocumentAbstractText, and/or DocumentFiledDate may be changed (in addition to any other Document metadata). Since the identification of the Document to be replaced is by DocumentIndexNumber, the AMEND-REPLACE directive must not be preceded by any other Transaction elements with directives that can affect index numbering (i.e. AMEND-INSERT, AMEND-REMOVE), either within a single Submission or within a group of Submissions. When this Transaction directive is used, the Transaction is limited to a single IndexofRecordDocument.

AMEND-REMOVE this Submission is altering an existing Record on Appeal by removing an existing Document. The receiving system is not obligated (upon acceptance) to delete the Document from the Document Management System (DMS) or Case Management System (CMS), although such a procedure is recommended. The receiving system is required to renumber all the Documents contained in the record, beginning with the Document which immediately followed the removed Document. Since the identification of the Document to be removed is by DocumentIndexNumber, the AMEND-REMOVE directive must not be preceded by any other Transaction elements with directives that can affect index numbering (i.e. AMEND-INSERT), either within a single Submission or within a group of Submissions. When this Transaction directive is used, the IndexOfRecordDocument is limited to a single Document and DocumentContent should be absent or empty.

AMEND-INSERT this Submission is altering an existing Record on Appeal by inserting a new Document between two existing Documents. The receiving system is required to renumber all the Documents contained in the record, beginning with the Document which immediately follows the inserted Document. Since the identification of the Document to be inserted is by DocumentIndexNumber, the AMEND-INSERT directive must not be preceded by any other Transaction elements with directives that can affect index numbering (i.e. AMEND-REMOVE), either within a single Submission or within a group of Submissions. When this Transaction directive is used, the IndexOfRecordDocument is limited to a single Document.

content identifies whether the Submission contains all of the Documents and DocumentContent necessary to complete the Transaction, or if the Transaction has been partitioned across multiple Submissions. Only two values are supported:

COMPLETE the Submission contains all the necessary documents and data to perform the Transaction. Whenever possible, COMPLETE should be used. When COMPLETE is used, and the IndexOfRecord contains a single Transaction, then the IndexOfRecord groupCount must have a value of “1”.

SEGMENTED the Transaction is not fully contained within the Submission, and multiple Submissions have been employed to transmit the full transaction content. The use of SEGMENTED Transactions should be avoided whenever possible. However, when initially filing a very large Record On Appeal, it may be necessary to use a group of Submissions, thereby necessitating the use of SEGMENTED Transaction content.

```
<!ELEMENT Transaction (IndexOfRecordDocument+)>
  <!ATTLIST Transaction type CDATA #FIXED “ROA”
                        id ID #REQUIRED
                        directive (FILE|AMEND-ADD|AMEND-REPLACE|AMEND-REMOVE|AMEND-INSERT) #REQUIRED
                        content  (COMPLETE|SEGMENTED) #REQUIRED >
```

IndexOfRecordDocument identifies a Document and its Index number on the Index of Record.

```
<!ELEMENT IndexOfRecordDocument (DocumentIndexNumber, Document)>
```

DocumentIndexNumber identifies the Document and the sequence number of the Document in the IndexOfRecord. Documents should be presented within the Transaction in ascending DocumentIndexNumber order. If the Submission is for an initial Record on Appeal filing (Transaction directive=FILE), then the first Document within the Submission (or the first Submission within a Submission group) must have a DocumentIndexNumber of “1” and each successive Document should have a DocumentIndexNumber one integer unit greater than the DocumentIndexNumber that immediately precedes it.

```
<!ELEMENT DocumentIndexNumber (#PCDATA)>
```

Document identifies a document within the Index of Record. A Document may be partitioned into pieces (typically a page, as in a single page TIFF document). An optional DocumentCommentText may be associated with the Document. Document has a single attribute:

id uniquely identifies a Document. When Documents are obtained from an OnBase DMS, then id should be the OnBase Document Handle value preceded by an underscore character.

```
<!ELEMENT Document (DocumentTypeText, DocumentSubtypeText?, DocumentTitleText,
                    DocumentFiledDate, DocumentAbstractText?, DocumentCommentText?,
                    DocumentPartition+)>
  <!ATTLIST Document id ID #REQUIRED>
```

DocumentTypeText identifies the classification of the Document at the SubmissionCourtName.

```
<!ELEMENT DocumentTypeText (#PCDATA)>
```

DocumentSubtypeText identifies a further sub classification of the Document at the SubmissionCourtName. This element need not be provided.

```
<!ELEMENT DocumentSubtypeText (#PCDATA)>
```

DocumentTitleText identifies the GIVEN title of the document (emphasis from Administrative Order 99-75). The DocumentTitleText is the name or title of the Document as provided by the Document's author or filer. On occasion, the DocumentTitleText may be the same as, or reflect the DocumentTypeText.

```
<!ELEMENT DocumentTitleText (#PCDATA)>
```

DocumentFiledDate identifies the date of filing in the SubmissionCourtName. Filing is generally understood to be the act of accepting the document by the court and officially including it in the case. This is not the receipt date of the Document by the SubmissionCourtName.

```
<!ELEMENT DocumentFiledDate (#PCDATA)>
        <!—Use ISO 8601 format, i.e. ccyy-mm-dd -->
```

DocumentAbstractText is a statement which summaries the content of a Document. This element is optional.

```
<!ELEMENT DocumentAbstractText (#PCDATA)>
```

DocumentSealedIndicator is a flag which identifies a document as sealed or not sealed. This element is optional. When not present, the document will be considered as not sealed. When used, and the document is to be considered as sealed, the value of the element should be “Yes”. All other values will be interpreted as not sealed.

```
<!ELEMENT DocumentSealedIndicator (#PCDATA) >
```

DocumentPartition identifies the portions into which a document’s contents are divided. Every Document must have at least one DocumentPartition. If a Document has only one DocumentPartition, then all of its content will be contained within the single partition. However, some documents are divided into many partitions that collectively make up the full document content. For example, some Document Management and Imaging systems, including OnBase, can internally store documents as an ordered collection of single pages. Each page represents a DocumentPartition. Extractor systems are free to combine multiple DocumentPartitions to form a single DocumentPartition.

```
<!ELEMENT DocumentPartition (DocumentFileName, DocumentPartitionNumber?, DocumentContent) >
```

DocumentFilename identifies the name of the file containing DocumentPartition content (or the full document content if there is single partition) at the SubmissionCourtName. This DocumentFilename need not be used by the receiving court which is free to use any convention for naming any necessary files. This element exists to facilitate compatibility with legacy transport mechanisms.

```
<!ELEMENT DocumentFileName (#PCDATA)>
```

DigestMethod identifies the algorithm used to produce the message digest for the document referenced by DocumentContent.

```
<!ELEMENT DigestMethod (#PCDATA)>
  <!ATTLIST DigestMethod algorithm CDATA #REQUIRED>
```

DigestValue the value, expressed in Base64 format, produced by applying the DigestMethod algorithm to the document content referenced by DocumentContent. This value, along with DigestMethod, can be used by the receiving application to verify the integrity of the document content in the external MQ messages.

```
<!ELEMENT DigestValue (base64)>
```

DocumentPartitionNumber identifies the order in which pages/parts of a partitioned document should be sequenced. Some imaging and document management systems are capable of partitioning documents into an ordered series of single pages. Values for DocumentPartitionNumber should be integer values. When a Document has a single DocumentPartition, then the DocumentPartitionNumber value may be either “0” or “1” (the value of “1” is recommended, but the value “0” is permitted to facilitate compatibility with legacy transport mechanisms). When a Document contains multiple partitions, then the value of DocumentPartitionNumber should begin with “1” and increase by a single integer value for each successive partition. DocumentPartitions should be placed into the E-ROA document in ascending DocumentPartitionNumber order.

```
<!ELEMENT DocumentPartitionNumber (#PCDATA)>
```

DocumentContent refers to the actual document; its visible content and any internal structures necessary to support the document. DocumentContent references all of the binary data (i.e. BLOB) that was contained in the DocumentFilename file at the SubmissionCourtName. The actual content is contained outside (i.e. external to) the e-ROA XML document and is referenced using XLINK.

id uniquely identifies DocumentContent.

mimeType specifies how the contents of the decoded PCDATA are to be interpreted (e.g., application/PDF).

size indicates the size, in bytes, of the referenced document content.

xmlns:xlink identifies the XLINK namespace and must be bound to the http://www.w3.org/1999/xlink namespace URI.

xlink:type identifies the XLINK element type as “simple”.

xlink:href identifies the location of the document content as a URI as defined in ‘IETF RFC 2396’ (see http://www.ietf.org/rfc/rfc2396.txt). The URI should reference the message which contains the document. This may be done by referencing the ‘Filename’ field of the AJIN MQ Message Header.

```
<!ATTLIST DocumentContent   id ID #REQUIRED
                            mimeType CDATA #REQUIRED
```

```
                                        size CDATA #IMPLIED
                                        xmlns:xlink CDATA #FIXED “http://www.w3.org/1999/xlink”
                                        xlink:type CDATA #FIXED “simple”
                                        xlink:href CDATA #REQUIRED >
```

SupplementalItems identifies other Documents that are transmitted along with the Record on Appeal. This element is optional. The rules of the court permit documents, other than those contained within the Record on Appeal, to be transmitted to the appellate court along with the record, however, this occurs infrequently. Supplemental documents must be associated with the SubmissionCourtName case. Under rare circumstances, an appellate court may order documents from a SubmissionCourtName case that is not a lineage predecessor of the appellate case. If this should occur, SupplementalItems should not be used to supplement the record. A new Submission should be used which identifies the appropriate SubmissionCourtName case from which the supplemental documents originate.

The action that the receiving court should take for any or all SupplementalItems is not defined. However, the DocumentCommentText element may be used to communicate processing instructions.

SupplemetalItems has one attribute:

docCount identifies the total number of SupplementalItems included in the Submission group.

```
<!ELEMENT SupplementalItems (DocumentCommentText?, Document+)>
   <!ATTLIST SupplementalItems   docCount CDATA #REQUIRED >
```

DocumentCommentText identifies narrative text relevant to the document. When included within the SupplementalItems element, DocumentCommentText can be used to clarify the nature of or the reason for the inclusion of the supplemental documents, as well as any processing expectations, or other miscellaneous information. When included with the Document element, DocumentCommentText can be used to provide additional notations, handling instructions, etc. and should be treated as temporary information.

```
<!ELEMENT DocumentCommentText (#PCDATA)>
```

Samples:

A simple initial ROA Filing, fully contained within a single Submission, transmitting two documents:

```
<?xml Version=1.0 encoding=”UTF-8” ?>
!--copyright @ 2005, Arizona Supreme Court. All Rights Reserved. -->

<!DOCTYPE E-ROA SYSTEM “e-ROA_submission.dtd”>
<Submission type=”ROA” version=”1.0” status=”CREATED”>
    <SubmissionInfo>
       <SubmissonCourt code=”S-1300”>Yavapai Superior Court</SubmissionCourtName>
       <SubmissionDateTime>2005-10-12T13:12:32.125</SubmissionDateTime>
       <SubmissionPersonName>Jane K. Doe</SubmissionPersonName>
       <SubmissionSeqNum>1</SubmissionSeqNum>
    </SubmissionInfo>
    <IndexOfRecord id=”IOR1” version=”1” groupCount=”1” docCount=”2”>
       <CaseNumber displayMask=”@@-##########”>CR0200400100</CaseNumber>
       <CaseTitleText>State v Burns</CaseTitleText>
       <AppellateCourtName code=”1 CA”>Court of Appeals Division One</AppellateCourtName>
       <AppellateCaseNumber></AppellateCaseNumber>
       <DocumentContent id=”IOR12345” mimeType=”application/pdf” size=”23477”
    xlink:href=”IndexofRecord.pdf” />
       <Transaction id=”Txn1” directive=”FILE” content=”COMPLETE”>
           <IndexOfRecordDocument>
             <DocumentIndexNumber>1</DocumentIndexNumber>
             <Document id=”_12345”>
               <DocumentTypeText>Rule 32 Petition</DocumentTypeText>
               <DocumentSubtypeText></DocumentSubtypeText>
               <DocumentTitleText>PETITION FOR POST CONVICTION RELIEF</DocumentTitleText>
               <DocumentFiledDate>2002-04-01</DocumentFiledDate>
               <DocumentAbstractText>Petitioner’s Rule 32 Petition for Post Conviction
    Relief</DocumentAbstractText>
               <DocumentPartition>
                 <DocumentFileName>Rule32Petitions.pdf</DocumentFileName>
                 <DigestMethod algorithm=http://www.w3.org/2000/09/xmldsig#sha1 />
                 <DigestValue>j61wx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
                 <DocumentPartitionNumber>1</DocumentPartitionNumber>
                 <DocumentContent id=”DC0001”
                                  mimeType=”application/pdf” size=”450945”
                                  xlink:href=”12345.pdf”</DocumentContent>
               </DocumentPartition>
```

```
            </Document>
          </IndexOfRecordDocument>
          <IndexOfRecordDocument>
            <DocumentIndexNumber>2</DocumentIndexNumber>
            <Document id="_23456">
              <DocumentTypeText>Rule 32 Notice</DocumentTypeText>
              <DocumentSubtypeText></DocumentSubtypeText>
              <DocumentTitleText>NOTICE OF POST CONVICTION RELIEF (PURSUANT TO RULE 32 OF THE RULES OF
    CRIMINAL PROCEDURE)</DocumentTitleText>
              <DocumentFiledDate>2002-04-01</DocumentFiledDate>
              <DocumentPartition>
                <DocumentFileName>Rule32Notices.pdf</DocumentFileName>
                <DigestMethod algorithm=http://www.w3.org/2000/09/xmldsig#sha1 />
                <DigestValue>UrXLDLBIta6skoV5/A8Q38GEw44=</DigestValue>
                <DocumentPartitionNumber>1</DocumentPartitionNumber>
                <DocumentContent id="DC0002"
                                 mimeType="application/pdf" size="105875"
                                 xlink:href="23456.pdf"</DocumentContent>
              </DocumentPartition>
            </Document>
          </IndexOfRecordDocument>
        </Transaction>
      </IndexOfRecord>
</Submission>
```

Frequently Asked Questions (FAQ)

1. How can I tell if a Submission is part of a group?

   When a Record on Appeal or a record alteration cannot be reasonably accommodated by a single Submission, then multiple Submissions may be used. The groupCount attribute of the IndexOfRecord element should be set to the total number of Submissions in the group. If this attribute value is greater than one, then the Submission is part of a group, otherwise the Submission is complete.

2. When a group of Submissions is used, how can I identify the Submissions which belong to the group?

   All of the Submissions which belong to the same Submission group will have the same values for:

   SubmissionCourtName
   SubmissionCourtName code
   CaseNumber
   AppellateCourtName
   AppellateCourtName code
   AppellateCaseNumber
   IndexOfRecord id

3. What is a good value for the id attribute of the IndexOfRecord element? Can I just use the sending Court's case number?

   The id attribute on the IndexOfRecord element is a necessary component to uniquely identify a Submission group (see FAQ 2). However, the idea of an IndexOfRecord id may seem confusing. The natural response would be to ID the IOR by CaseNumber (or possibly Court and CaseNumber). This approach will work when a single Submission is required to transfer the record. However when multiple Submissions are required, use of CaseNumber will not meet the IndexOfRecord uniqueness constraint, necessary to permit Submission group identification.

4. For a group of Submissions, how can I tell the order in which the Submission should be handled?

   Each Submission within the group will have a SubmissionSeqNumber. The SubmissionSeqNumber of the first Submission in the group should be "1". The SubmissionSequenceNumber of the second Submission within the group should be "2", etc., until the last Submission, which should have a SubmissionSeqNumber equal to the IndexOfRecord groupCount. Submissions should be handled in ascending SubmissionSeqNumber order.

5. When a Submission group is used, how can I identify the documents within the record which are transmitted within a given Submission within the Submission group?

   Each Submission within the Submission group will contain an e-ROA XML document which defines the Submission. The IndexOfRecord element within each Submission will identify all the Documents of the record which are contained within the Submission.

   This implies that the IndexOfRecord element for a given Submission within a Submission group will only list the documents that it carries (like a packing slip).

6. When a Submission Group is used, how can I identify all the documents which comprise the complete Record On Appeal?

   The full Record on Appeal can be identified from the IndexOfRecord DocumentContent. However, since this may be in human readable form, if the Transaction directive is FILE, software applications may prefer to identify all the documents by concatenating the IndexOfRecordDocuments for all the Submissions within the group. When the Transaction directive is other than FILE, then concatenating the IndexOfRecordDocuments will not identify all the documents which comprise the full Record On Appeal. This information will need to be obtained from the receiving court's EDMS or CMS.

7. How can I determine the number of documents that should be included within a Submission group?

   A Submission group, whether it contains a single Submission or multiple Submissions can contain documents from three different 'sources' (not counting the E-ROA XML documents). These are: (1) the Index of Record document, (2) the Index of Record content documents (i.e. the record), and (3) Supplemental Item documents.

   Index of Record document – there should be one and only one Index of Record document, no matter how many Submissions are contained within the Submission group, and no matter how many Transactions are contained within the Submission group.

   Index of record content – the number of documents contained as part of the Index of Record content can be determined from the docCount attribute of the IndexOfRecord element. If there are multiple Submissions within the Submission group, then the IndexOfRecord docCount will need to be summed for all Submissions within the group.

   Supplemental Items – the number of Supplemental Item documents can be determined from the docCount attribute of the SupplementalItems element. If there are multiple Submissions within the Submission

group which contain SupplementalItems, then the SupplementalItems docCount will need to be summed for all relevant Submissions within the group.

e.g. Total Documents = 1 + IndexofRecord.docCount + SupplementalItems.docCount

8. How can I tell if a document has been completely received and is intact?

If the Document has been provided within a single DocumentPartition, then:

Normally a document will be sent in a single logical MQ message. However, if the document is large, it may require multiple physical MQ messages to transmit. Due to physical constraints (such as queue size, or buffer size, etc.), MQ may divide the logical message into multiple physical messages. This is called arbitrary segmentation. If MQ performs arbitrary segmentation, then the sending application will not even be aware of the segmentation. As such, the sending application would not be able to set any type of status or flag indicating segmentation. However, arbitrary segmentation is visible to the receiving application (i.e. MQReader). By setting MQGMO_ALL_MSGS_AVAILABLE, the receiving application can assure that it will not retrieve a message until all messages or segments within a group are available in the queue.

If a receiving application were to retrieve segments individually from the queue, then the size attribute on the DocumentContent element can be used to determine when the document content is complete.

If MQ is used, then the MD Offset field can be used to determine how much of the document content has been transmitted. When the segment's Offset + OriginalLength value equals the DocumentContent size attribute, then the document has been fully received. Note: the OriginalLength field refers to the length of the original physical message, not the logical message!

If a DigestMethod and DigestValue have been provided, then the document content can be 'hashed' using the DigestMethod and the result compared to DigestValue. If they match, then the document is intact and unaltered.

If the document spans multiple DocumentPartitions, then each DocumentPartition will need to be evaluated independently (as described above for a single DocumentPartition). When all DocumentPartitions for the Document have been fully received, then the Document will have been fully received.

9. When an excessively large document gets split into multiple MQ messages, how are all the related messages identified? Can a field in the MD be used for this purpose? Should there also be information in the XML that indicates this?

   Currently there is no indicator value or flag in the XML that identifies that a document has been split into multiple logical messages. The receiving application could infer this by (1) observing that there are more logical messages that would be expected by the information contained within the XML, and (2) by comparing the DocumentContent size attribute value to the number of bytes read from the MQ payload. If the size is not accommodated by message payload, then the document must be continued in the next logical message (or there is an error).

10. When an acceptance or rejection message is received by the sending court, how will they know to which Submission or Submission group the response message refers?

    MQ documentation suggests sending the original message's MsgId as the responding messages' CorrellationId. MsgId and CorrellationId will be passed to fielders from MQReader. Applications should retain values until any need to reply has lapsed. For ROA transactions, the receiving appellate court can reject the submission. The rejection notification should identify the transaction being returned.Although IndexOfRecord ID could be considered for this purpose, it is more customary in an MQ environment to return the MsgId of the transmitted message as the CorrellationId of the rejection message. There may be considerable delay (i.e. hours) between receipt of an ROA transaction and its rejection or acceptance. If MsgId is to be returned, it must persist throughout this delay interval.

11. Why was document content not included within the XML document in Base64 format as defined by LegalXML CourtFiling 1.1?

    This approach was initially considered. However, since a record on appeal can contain many (up to hundreds) of documents, this would have produced an excessively large XML document (i.e. file). Except in the rarest of circumstances (i.e. a small record), this XML document would exceed the size limits of an MQ message and need to be 'split' across multiple messages. This would be needlessly complicated. Instead, it was decided that each document should be transported in its own individual message (when possible).

12. Since each document will be passed in MQ on separate messages, how should the XML in the first MQ message, reference the MQ message which contains the document content? Is it better to use unparsed external entities or is XLINK a better choice?

Through the use of external unparsed entities and notations, non-XML data can be ‘included’ in the XML document. The entity declaration identifies the location of the external content (i.e. a URL) and references a notation. The notation identifies the format of the external content (e.g. MIME type). The XML fragments below demonstrate the use of an unparsed external entity for external content ‘inclusion’:

```
<!ENTITY DOC1 SYSTEM “filename1.ext” NDATA TIFF>
<!NOTATION TIFF SYSTEM “image/tiff”>

<!ELEMENT DocumentContent EMPTY>
<!ATTLIST DocumentContent   id ID #REQUIRED
                            size CDATA #IMPLIED
                            source ENTITY #REQUIRED >


<DocumentContent id=_12345 size=23987 source=”DOC1” />
```

An alternative to unparsed external entities is to use XLINK. XLINK is the XML way of providing document references, typically for the purpose of hyper-linking. The following fragments demonstrate the use of XLINK:

```
<!ELEMENT DocumentContent EMPTY>
<!ATTLIST DocumentContent   id ID #REQUIRED
                            size CDATA #IMPLIED
                            xmlns:xlink CDATA #FIXED “http://www.w3.org/1999/xlink”
                            xlink:type CDATA #FIXED “simple”
                            xlink:href CDATA #REQUIRED
                            mimeType CDATA #REQUIRED >

<!DocumentContent id=_12345 size=23987 xlink:href=”filename1.ext” mimeType=”application/tif”>
```

Pros and Cons:

Browser support is better for unparsed external entities than for XLINK. However, since the e-Appeal usage is for data/document transfer and not display, this is of little concern. However, there is interest in using XSL style sheets to present the e-ROA XML as an Index of Record document.

Overall, the XLINK approach has been selected because it appears simpler than the external unparsed entity approach.

13. For MQ messages which contain document content, should each message contain at most a single document? Is there a practical limit on the size of a document that can be allowed into a single message? Should a single document be allowed to be split across multiple MQ messages? Should MQ arbitrary segmentation be permitted?

    First, it should be noted that MQ supports two forms of segmentation (dividing a logical message into multiple physical MQ messages). 'Arbitrary' segmentation is when MQ divides large messages for transport. Arbitrary segmentation can be invisible to the applications using MQ. When the receiving application GETs a message an application has the option to obtain the full message PUT initially, or obtain each segment individually. 'Application' segmentation is when the sending application divides a 'logical' message into multiple parts for transport. Both arbitrary and application segmentation may be used together, these features are not mutually exclusive.

    Secondly, it should be noted that 'segmentation' is different than 'grouping'. MQ permits groups of messages to be established and sent. The receiving application can specify that it should not be notified until all messages within the group have been received (MQGMO_ALL_MSGS_AVAILABLE).

    It seems reasonable to place each document into a single MQ message by itself. A single MQ message cannot be larger than 100MB (104,857,600 bytes) which includes the MD. This should almost always be large enough for a single document. By permitting MQ to perform arbitrary segmentation, it should not be necessary to require the application to split documents into multiple message segments. Applications will need to split documents that exceed the single message size limitation (i.e. 100MB) into multiple messages. Additionally, applications must guard against exceeding the size limitations of a message Group (limited to approximately 1 GB). When the space remaining in a message Group is smaller than the size of the next Document to send, then the Submission must be terminated, and the next Submission within the Submission group begun.

14. Some elements such as DocumentAbstractText, DocumentCommentText, DocumentSealedIndicator, DigestMethod, and DigestValue are optional. Why are optional elements included? When left blank (i.e., no value provided) do empty tags need to be included in the XML?

    Due to the varying case and document management systems used throughout the state, flexibility is essential. This Information Exchange Package (IEP) has not only been designed to accommodate state standard systems such as OnBase, AZTEC, and Appellamation, but also various other case and document management systems in use throughout the state. Different systems, and different implementations of systems, support different capabilities. For example, many document management systems support not only Document Titles, but also Document Abstracts. When Abstract information is available, there should be a way to send it to a receiving system which may also benefit from the available information. The element is optional so that more limited EDMS's which do not support Document

Abstract information, are not required to send it. The inclusion of optional elements is consistent with the LegalXML philosophy of “inclusive and optional.”

Optional elements which have no values are not required to be included in the XML. The ‘?’ in the DTD (as in: <!ELEMENT Document (DocumentTypeText, DocumentSubtypeText?, DocumentTitleText, DocumentFiledDate, DocumentAbstractText?, DocumentCommentText?, DocumentSealedIndicator?)>) specifies that the element is optional, and should occur once or not at all. For example, the following XML fragments are valid, based on the DTD fragment above:

```
<Document id=”_134529”>
      <DocumentTypeText>Notice of Appeal</DocumentTypeText>
      <DocumentTitleText>Motion to Appeal</DocumentTitleText>
      <DocumentFileDate>2004-05-23</DocumentFiledDate>
</Document>

<Document id=”_1708”>
      <DocumentTypeText>Letter</DocumentTypeText>
      <DocumentTitleText>Austin, Gary - Letter of 10 Sept 1997</DocumentTitleText>
      <DocumentAbstractText>Requesting Info on SA - See Rules on SA</DocumentAbstractText>
      <DocumentFileDate>1997-09-12</DocumentFiledDate>
</Document>

<Document id=”_145705”>
      <DocumentTypeText>Petition</DocumentTypeText>
      <DocumentTitleText>Petition for Allowance and Payment of Claim</DocumentTitleText>
      <DocumentAbstractText></DocumentAbstractText>
      <DocumentFileDate>2005-12-22</DocumentFiledDate>
      <DocumentSealedIndicator>Yes</DocumentSealedIndicator>
</Document>
```

Issues:

1. How should the DocumentContent URI be defined?

   A Uniform Resource Identifier (URI) is a compact string of characters for identifying an abstract or physical resource. A URI can be further classified as a locator (URL), a name (URN), or both.

   For the purpose of this E-ROA XML, we want to use a locator type (URL) that will identify the actual location of the external document content.

   If the XML is to be 'valid' at all points within its lifecycle, then this URL will need to be revised as the message is first created, then transmitted, then received. When first created by an Assembler, or after receipt by an MQReader, the URL would point to a file which contained the document content. If the file is located in the same directory as the XML document, then a relative URL should be used. When being transported using MQSeries, the document content will exist in an MQ message, and the URL should identify the message. This task is further complicated by the possibility of multiple Submissions (i.e. a Submission group) being required to transport the complete ROA.

   How important is it for the state of the XML to be always valid?

2. Is support for Volumes required?

   Currently, with paper records, the trial court will sometimes bind up multiple documents within the record into bound volumes. This is presumably done to facilitate handling and indexing of these records. Although this seems unnecessary with electronic documents, will courts continue to define volumes? Would an optional volumeNumber attribute on the Document element suffice?

   It is currently understood that Volume support is not necessary.

3. Should MQ Server binding or MQ Client binding be used?

   When using Server binding mode, WebSphere MQ classes use the queue manager API, rather than communicating through a network. This provides better performance for WebSphere MQ applications than using network connections.

The MQ Queue Manager's *Begin* method for .NET is only available in Server Binding mode.

In the 'MQSeries Version 5 Programming Examples' (SG24-5214) it advises that for persistent messages, the queue manager can reassemble segments only within a unit of work.

Is the *Begin* method required to start a unit of work?

It appears that a Begin method is only required to start a Global Unit of work, but not a Local unit of work (see MQSeries Application Programming Guide SC33-0807-12, page 183).

4. Should the MQReader wait for an affirmation from a Fielder before causing messages to be deleted from a queue (by committing; MQCMIT)? If messages are deleted from the MQ queue before being committed by the Fielder, how can they be recovered? Should two phased commit be used? Should MQReader be an 'external synch point coordinator'?

   These issues are being addressed by the MQReader expansion team. The interface between MQReader and Fielders specifies that when processing an MQ Group, MQReader will not commit until the whole group has been read, passed to the Fielder, and the Fielder signals to MQReader that it is okay to commit. If the Fielder signals MQReader that there is a problem, the Fielder will indicate the problem severity. If the problem is correctable, then MQReader may rollback, leaving the message group in the queue for correction and subsequent processing. More likely however, is that the problem cannot be easily corrected and the errant message must be removed from the queue and placed into a holding area for resolution.

5. Should a Document be permitted to have multiple DocumentTypeText elements?

   Many documents cannot be fully described by a single document type classification, and many Document Management Systems support multiple document type assignments for a single document.

6. The E-ROA XML specification supports a DocumentSealedIndicator. Is this enough? Is there a need to indicate that a document is 'Confidential' but not 'Sealed'? If so, what is the difference?

   Sometimes the 'rules' contain language that suggests that there is a difference between 'sealed' document and a 'confidential' document. For example, from the Code of Judicial Administration 1-506 E 4:

   "Filing of confidential and sealed documents. Courts shall not accept electronically filed confidential and sealed documents."

Even though the electronic transfer of the record is a form of electronic filing (court to court e-filing), is it correct to interpret that the above constraint does not apply to the e-filing of the record?

7. Reserved for future issues.