OASIS Committee Note

Implementation Guidance for Electronic Court Filing Version 4.1

Committee Note 01

16 October 2023

This stage:
https://docs.oasis-open.org/legalxml-courtfiling/ecf-guide/v4.1/cn01/ecf-guide-v4.1-cn01.docx (Authoritative)
https://docs.oasis-open.org/legalxml-courtfiling/ecf-guide/v4.1/cn01/ecf-guide-v4.1-cn01.html
https://docs.oasis-open.org/legalxml-courtfiling/ecf-guide/v4.1/cn01/ecf-guide-v4.1-cn01.pdf

Previous stage:
N/A

Latest stage:
https://docs.oasis-open.org/legalxml-courtfiling/ecf-guide/v4.1/ecf-guide-v4.1.docx (Authoritative)
https://docs.oasis-open.org/legalxml-courtfiling/ecf-guide/v4.1/ecf-guide-v4.1.html
https://docs.oasis-open.org/legalxml-courtfiling/ecf-guide/v4.1/ecf-guide-v4.1.pdf

Technical Committee:
OASIS LegalXML Electronic Court Filing TC

Chair:
James Cabral (jim.cabral@infotrack.com), InfoTrack US

Editors:
James Cabral (jim.cabral@infotrack.com), InfoTrack US
Gary Graham (GGraham@courts.az.gov), Arizona Supreme Court

Additional artifacts:
This document is one component of a Work Product that also includes:
Schema and WSDL files from *Electronic Court Filing Version 4.1:* https://docs.oasis-open.org/legalxml-courtfiling/ecf-guide/v4.1/cn01/ecf-4.1-wssip-adapted-to-4.01/.

Related work:
This document is related to:

- *Electronic Court Filing Version 4.1.* Edited by James Cabral, Gary Graham, and Philip Baughman. Latest stage: https://docs.oasis-open.org/legalxml-courtfiling/ecf/v4.1/ecf-v4.1.html.

Abstract:
This committee note provides non-normative guidance to implementers of the LegalXML Electronic Court Filing Version 4.1 specification.

Status:
This is a Non-Standards Track Work Product. The patent provisions of the OASIS IPR Policy do not apply.

This document was last revised or approved by the OASIS LegalXML Electronic Court Filing TC on the above date. The level of approval is also listed above. Check the "Latest stage" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=legalxml-courtfiling#technical.

TC members should send comments on this document to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/legalxml-courtfiling/.

Citation format:
When referencing this document, the following citation format should be used:

[ECF-Guide-v4.1]

Implementation Guidance for Electronic Court Filing Version 4.1. Edited by James Cabral and Gary Graham. 10 October 2023. OASIS Committee Note 01. https://docs.oasis-open.org/legalxml-courtfiling/ecf-guide/v4.1/cn01/ecf-guide-v4.1-cn01.html. Latest stage: https://docs.oasis-open.org/legalxml-courtfiling/ecf-guide/v4.1/ecf-guide-v4.1.html.

Notices:

Table of Contents

1 Introduction

This document is a Technical Committee Note and is not a Specification document. Since this document is not a specification, any inadvertent discrepancies or contradictions are governed by the normative requirements of the actual specification and specification documents.

This document provides implementation guidance for the **[ECF-v4.1]** and **[ECF-WS-SIP-v4.1]** specifications. Since the Technical Committee (TC) anticipates that implementors of the v4.1 specifications will have previously implemented prior ECF versions, especially the **[ECF-v4.01]** and related errata and Web Services SIP specifications. As such, this document will identify differences between the **[ECF-v4.1]** and **[ECF-WS-SIP-v4.1]** specifications and immediate prior ECF specifications.

This document will also address other implementation issues. The TC may use this Committee Note to provide specification clarifications and to make non-normative suggestions or recommendations to implementors.

1.1 Changes from earlier Versions

Changes to this document are tracked in Appendix E. Revision History.

1.2 Glossary

1.2.1 Definitions of terms

This document incorporates the defined terms in **[ECF-v4.1]** and **[ECF-WS-SIP-v4.1]** by reference.

This section defines additional key terms used in this Committee Note:

ECF Message

An XML instance of one of the XML structures defined by schema provided in the ECF provided `xsd/message` folder. ECF provides 24 message structures. Although ECF Messages are specified as MDE operation parameters, ECF Messages are not provided directly to operations. The ECF Messages are wrapped within request and response structures for exchanges.

Exchange

A communication between MDEs initiated by means of an XML instance provided as an invocation parameter to an MDE operation and resulting in a synchronous XML response.

Submission

An individual e-filing tendering provided by the FAMDE to the FRMDE as a single transaction (e.g., by a single invocation of the ReviewFiling operation); i.e., a `ReviewFilingRequest`.

Acronyms and abbreviations

CRMDE	Court Record Major Design Element
ECF	Electronic Court Filing – this acronym generally refers to the OASIS LegalXML Electronic Court Filing Technical Committee or one of this committee's specifications.
FAMDE	Filing Assembly Major Design Element
FRMDE	Filing Review Major Design Element
MDE	Major Design Element
MRM	MessageReceiptMessage
NIEM	National Information Exchange Model
RFR	RecordFilingRequest
RvFR	ReviewFilingRequest
SIP	Service Interaction Profile
TC	Technical Committee - this acronym refers to the OASIS LegalXML Electronic Court Filing Technical Committee
URI	Uniform Resource Identifier

1.2.2 Document conventions

- Naming conventions
- Font colors and styles
- Typographical conventions

- XML element and attribute names are displayed in Courier New font, e.g., `nc:DocumentIdentification`

- Literal quotations are contained with double-quote characters and are highlighted using blue text.

2 Installation

This section provides suggestions for the installation of the files provided in the **[ECF-v4.1]** and **[ECF-WS-SIP-v4.1]** specifications.

2.1 Environment

A working **[ECF-v4.1]** environment permits review of the specification and technical artifacts. This section provides instructions for setting up a working **[ECF-v4.1]** environment.

Implementors may choose to alter the locations of files and/or use alternative folder structures and names in an actual production environment. Before making this choice, it should be noted that the various ECF artifacts are deployed using relative folder path references. Revisions to ECF provided artifacts may be necessary if the production environment is different that the **[ECF-v4.1]** working environment.

2.1.1 Relative Paths and schemaLocation

ECF uses relative path designators in artifacts provided with specifications, typically for `import` and `schemaLocation`.

In doing so, all artifacts provided with the Core specification within the deployment zip file are properly located, one to another, within the zip file. This makes setting up a working **[ECF-v4.1]** environment, based on the Core specification, relatively easy.

However, when including Profiles, such as the **[ECF-WS-SIP-v4.1]** or Signature Profile specifications, then additional steps may be required.

2.1.2 Working Environment Set-up

By following the following instructions provided, ECF artifacts will be placed into proper folder locations so that the relative paths provided are correctly positioned. As such, artifacts will properly reference one another (e.g., in `schemaLocation`, etc.).

To set-up a working **[ECF-v4.1]** environment:

1. Determine a root folder location
2. Unzip (e.g., Extract All) from the Core specification zip file into your root folder
    a. When unzipping, remove the rightmost folder name from the suggested path.
3. Unzip (e.g., extract) the corresponding Web Services SIP files into the root folder
    a. Once unzipped, then cut the wsdl folder and paste it beneath the Core specification root folder at the same folder level as the xsd folder.

3 Backward Compatibility

This section addresses the differences between this version of ECF and the prior version with the same major version number, and backward compatibility implications arising from these differences.

Section 1.2 Relationship to Prior Specifications of the **[ECF-v4.1]** specification includes:

"This specification does not assume that prior specifications will be deprecated. However, ECF 4.1 is not guaranteed to be backward-compatible with previous versions including ECF 4.0 and 4.01, both based on NIEM 2.x. Applications based on ECF versions which themselves are based on NIEM versions other than NIEM 2.x (such as ECF 3.0, 3.01 and 3.1 specifications) will certainly not interoperate successfully with applications using this specification. This fact is indicated by the assignment of a new major and minor version number to the specifications."

In the OASIS **[Interoperability Guidelines]**, "backward compatibility" is defined as:

"Backward compatibility: A standard is said to allow backward compatibility, if products designed for the new standard can receive, read, view or process older standards or formats. Or, it is able to fully take the place of an older product, by inter-operating with products that were designed for the older product."

This section addresses the backward compatibility of the **[ECF-v4.1]** and **[ECF-WS-SIP-v4.1]** specifications with respect to the **[ECF-v4.01]**, **[ECF-v4.01-errata02]** and **[ECF-v4.0-WS-SIP-v2.01]** specifications.

3.1 Portable Media Messaging Profile Deprecated

[ECF-WS-SIP-v4.1] deprecates the use of the **[Portable Media Messaging Profile]**.

3.2 New Web Services SIP

[ECF-v4.1] is designed for compatibility with **[ECF-WS-SIP-v4.1]** . Prior Web Services SIP versions are not compatible with **[ECF-v4.1].**

Prior versions of the ECF Web Services SIP specification (e.g., **[ECF-v4.0-WS-SIP-v2.01]**) provided a single WSDL file for use for all MDEs and all MDE operations. **[ECF-WS-SIP-v4.1]** provides a separate WSDL file for each MDE.

For instance, while **[ECF-v4.0-WS-SIP-v2.01]**) declares a single XML namespace

- `urn:oasis:names:tc:legalxml-courtfiling:schema:xsd:WebServicesProfile-2.0`

and the namespace URI is also used as the Service Interaction Profile Identifier as specified in Section 2.1 'Service Interaction Profile Identifier', **[ECF-WS-SIP-v4.1]** declares four (4) namespaces:

- `urn:oasis:names:tc:legalxml-courtfiling:schema:wsdl:CourtRecordMDE-4.1`
- `urn:oasis:names:tc:legalxml-courtfiling:schema:wsdl:FilingAssemblyMDE-4.1`
- `urn:oasis:names:tc:legalxml-courtfiling:schema:wsdl:FilingReviewMDE-4.1`
- `urn:oasis:names:tc:legalxml-courtfiling:schema:wsdl:ServiceMDE-4.1`

and a single separate URI is used as the Service Interaction Profile Identifier:

- `urn:oasis:names:tc:legalxml-courtfiling:schema:xsd:WebServices-4.1`

[ECF-WS-SIP-v4.1] incorporates the new wrappers.xsd schema from the **[ECF-v4.01]** specification.

3.3 Bulk and Batch Filing Terminology

[ECF-WS-SIP-v4.1] no longer claims to support 'bulk filings'. The ECF TC has elected to discontinue use of the terminology of "Bulk filings" and "Batch filings" as, at this time, there does not appear to be industry/legal community-wide consensus on the definition of these terms.

3.4 Court Policy Changes

There are minor changes to Court Policy.

3.4.1 Machine-Readable Court Policy

[ECF-v4.1] makes it clear that some form of machine-readable court policy is required in a complete implementation. However, it is not necessary that machine-readable court policy is implemented in an ECF suggested manner, e.g., by providing the GetPolicy operation that utilizes the ECF-4.1-`CourtPolicyQueryMessage` and `CourtPolicyResponseMessage`.

When a non-ECF machine-readable court policy is implemented, the Core specification provides broad implementor discretion. However, implementors should be mindful of code-lists detailed in Section 2.4.4 'Court-specific Code Lists'.

3.4.2 CourtPolicyResponseMessage

Section 2.1 of **[ECF-WS-SIP-v4.1]** was corrected to specify the `CourtPolicyResponseMessage` and not the errantly specified `CourtFilingResponseMessage`.

3.4.3 RequireAsynchronousResponsesIndicator

The element `<RequireAsynchronousResponsesIndicator>` was added to `CourtPolicyResponseMessage` and to the machine-readable court policy.

When this element is 'false' then asynchronous callback messages need not be provided for any NotifyDocketingComplete operation or for any NotifyFilingReviewComplete operation. Take note that **[ECF-v4.1]** does not require suppression of both the `RecordDocketingCallbackMessage` elements and `ReviewFilingCallbackMessage/PaymentReceiptMessage` elements when `<RequireAsynchronousResponsesIndicator>` is 'false'. An implementation may choose to suppress `RecordDocketingCallbackMessage` elements while continuing to provide `ReviewFilingCallbackMessage/PaymentReceiptMessage`.

Since ECF machine-readable Court Policy is optional (although some form of machine-readable court policy is mandatory) then the `<RequireAsynchronousResponsesIndicator>` element may not be relevant. When the `<RequireAsynchronousResponsesIndicator>` element is not used, then this element should not be understood to be 'false' nor should it be presumed to be 'true'. In this circumstance, it is recommended that the presumed value should be stated in Human-readable court policy.

When `<RequireAsynchronousResponsesIndicator>` is 'true', then `<SendingMDELocationID>` and `<SendingMDEProfileCode>` MUST be included (with appropriate values) in all messages that provide these elements, such as `CoreFilingMessage`, `CaseListQueryMessage`,

`CourtPolicyQueryMessage`, `FeesCalculationQueryMessage`, `RecordDocketingCallbackMessage`, etc.

3.5 New Wrappers Schema

[ECF-v4.1] now includes the optional wrappers.xsd schema. However it is required when also using **[ECF-WS-SIP-v4.1]**.

Wrappers.xsd introduces request and response structures. These request and response structures fill a gap that has existed between the Core specification and the Web Services SIP specifications in prior ECF versions. This gap, by necessity, was typically filled in implementations by defining and using implementation specific exchange schema or by WSDL modifications or extensions.

One consequence is that terminology, such as `ReviewFilingRequest`, now has a more specific technical meaning in **[ECF-v4.1]**. (i.e., `ReviewFilingRequest` is a complex element that contains one-to-many `CoreFilingMessage` elements and zero-to-one `PaymentMessage` element as defined in wrappers.xsd).

However, even though 'request' and 'response' structures are defined in wrappers.xsd, as 'types' (e.g., '`GetPolicyRequestType`'), there may not be corresponding 'Request' named elements for these types. However, all elements derived from response type structures are named such that the element name ends with 'Response'.

For example, although there is a `GetPolicyRequestType` defined in wrappers.xsd, there is not a `GetPolicyRequest` element defined in wrappers.xsd or in any other schema or WSDL. Instead, the `GetPolicy` element is derived from `GetPolicyRequestType`. However, wrappers.xsd defines both a `ReviewFilingRequestType` element and a `ReviewFilingRequest` element derived from corresponding type structures.

Also, one request type structure was named without including 'Request' within the type-name (i.e., `GetDocumentType` would be better named as `GetDocumentRequestType`). The corresponding operation response type structure (i.e., `GetDocumentResponseType`) is the basis for the response element `DocumentResponseMessage`.

Appendix C includes a list of all request and response type structures defined in wrappers.xsd.

3.6 Normative Operations Signatures

In prior ECF Core specifications, MDE operation signatures were not normatively defined, however 'suggested' (e.g., informative) operations signatures were provided in an Appendix (as Appendix C in prior specification documents). In **[ECF-v4.1]**, operation signatures, provided in Section 5 'MDE Operations', are normative. These operation signatures define the ECF message parameters, the cardinality for the parameters, and the order in which the parameter messages must be provided. The response output message is also specified.

[ECF-v4.01] implementations that support ECF operations with input and output parameters that are not conformant with the signatures defined in Section 5 will require modifications to those operations for ECF v4.1 compatibility.

If **[ECF-v4.01]** implementations follow the informative guidelines provided in Appendix C of the ECF **[ECF-v4.01]** specification, then these prior informative operation signatures have been replaced by normative operation signatures that are largely consistent with those of the prior ECF version.

Some **[ECF-v4.1]** operation signatures have been modified to relax cardinality restrictions and allow multiple ECF message parameters.

Since these cardinality changes are optional, implementations that do not take advantage of these newly allowed multiplicities are not expected to experience any issues. However, existing implementations that have exercised implementation/court-specific extension options, may require modifications. These local extensions should be reviewed. If the court/implementation extension was made to provide the same relaxed parameter cardinalities as provided in **[ECF-v4.1]** , then the court/implementation could retire that extension.

3.6.1 Relaxed Operation Signature Cardinality

Operation signature parameter cardinalities have been relaxed for some operations. Specifically:

1) `ReviewFiling` now permits multiple `CoreFilingMessage` elements.
2) `RecordFiling` now permits multiple `CoreFilingMessage` elements.
3) `NotifyDocketingComplete` now permits multiple `RecordDocketingCallbackMessage` elements. Additionally, `RecordDocketingCallbackMessage` may now contain multiple `ReviewedLeadDocument` elements.
4) `NotifyFilingReviewComplete` now permits multiple `ReviewFilingCallbackMessage` elements. Additionally, `ReviewFilingCallbackMessage` may now contain multiple `ReviewedLeadDocument` elements.
5) `FeesCalculationQueryMessage` now permits multiple `CoreFilingMessage` elements.

Although operation signatures are defined using ECF message structures as parameters, these normative parameters are bound within a request structure defined within wrappers.xsd.

Future Service Interaction Profile specifications compatible with **[ECF-v4.1]** may or may not employ the request and response structures defined in wrappers.xsd. For example, if/when the IBM MQ Service Interaction Profile is updated, this specification may or may not incorporate wrappers.xsd. However, the operation signatures and cardinalities provided in section 5 ‘MDE Operations’ must be adhered to.

3.7 Optional Callback Messages

Asynchronous callback messages are now optional for:

- `NotifyDocketingComplete`
- `NotifyFilingReviewComplete`

The TC observed that some courts prefer ‘auto-clerk-review’ acceptance of e-filing submissions for some or all matter types. ‘Auto-Accepted’ submissions are not reviewed by a clerk, and are directly docketed into the CRMDE, typically from the `ReviewFilingRequest` and not through a `RecordFilingRequest`.

3.8 Required and Optional Operations

The NotifyDocketingComplete operation and the NotifyFilingReviewComplete operation are no longer required operations. Also, depending upon the implementation, RecordFiling may also not be required.

However, when the <RequireAsynchronousResponsesIndicator> element in machine-readable Court Policy is ‘true’ and when the implementation is using the RecordFiling operation (see 3.1.8.3) then the NotifyDocketingComplete operation must be invoked.

Additionally, when the <RequireAsynchronousResponsesIndicator> element in machine-readable Court Policy is ‘true’ then the NotifyFilingReviewComplete operation must be invoked.

3.8.1 NotifyDocketingComplete

The NotifyDocketingComplete operation is now optional in **[ECF-v4.1]. I**mplementations that do not employ the RecordFiling operation (e.g., in an ECF conformant manner) also need not invoke the NotifyDocketingComplete operation. Additionally, implementations in which <RequireAsynchronousResponsesIndicator> is ‘false’ also need not invoke NotifyDocketingComplete.

For clarity, it should be noted that the NotifyDocketingComplete operation is a required operation in prior ECF versions, even when a submission is fully rejected in Clerk Review (see **[ECF-v4.01]** Section 3.2.7 ‘NotifyDocketingComplete’ which states: “The Court Record MDE MUST invoke the NotifyDocketingComplete operation on the Filing Review MDE as a callback message to the RecordFiling operation to indicate whether the filing was accepted or rejected by the court record system. If the Court Record MDE rejected the filing, an explanation MUST be provided)”.

How filing acceptance or rejection, and other docketing information is communicated to the Filing Review MDE is unspecified. The Core specification is silent on this. But it seems clear that if the clerk review results are to be made available to the FRMDE, then this information must be provided by means other than through the NotifyDocketingComplete operation, when the NotifyDocketingComplete operation is not used.

3.8.2 NotifyFilingReviewComplete

As stated previously, the NotifyFilingReviewComplete operation is now optional.

When the NotifyFilingReviewComplete operation is not utilized, then providing clerk review results, payment receipt information, and docketing information is challenging. This is considered next:

3.8.2.1 Providing Clerk Review and Other Results

NotifyFilingReviewComplete provides clerk review results, docketing results, and payment receipt information to the FAMDE. So, when NotifyFilingReviewComplete is not utilized then how does this information get communicated back to the FAMDE?

[ECF-v4.1] is silent on this. But it seems clear that if the clerk review results are to be made available to the FAMDE, then this information must be provided by means other than through the NotifyFilingReviewComplete operation, when the NotifyFilingReviewComplete operation is not used.

It may be notable to observe that, although implied, **[ECF-v4.1]** is not clear whether it is a requirement that the clerk review results for a submission must be communicated to the FAMDE, but this is a recommended practice. Section 2.2 ‘Major Design Elements’, bullet 2 ‘Filing Review MDE’ states that FilingReview “enables a court to receive and review a filing message and prepare the contents for recording in its case management and document management systems, sending a response concerning the filing to the Filing Assembly MDE”. The words “sending a response” at a minimum implies or suggests that the NotifyFilingReviewCompleteRequest should be sent.

One option for providing clerk review results to the FAMDE, when not providing `NotifyFilingReviewCompleteRequest`, is through the use of the GetFilingStatus operation. GetFilingStatus can return a single `FilingStatusCode` (e.g., 'received', 'accepted', 'partially-accepted', or 'rejected'). GetFilingStatus can also return limited docketing information in `nc:CaseDocketID` and `nc:Case`. Additional case and docketing information may be obtained using the GetCase operation. There are no ECF message queries that will provide payment receipt information.

If the `GetFilingStatus` request specifies a single 'filing' (e.g., by providing a 'filing-identifier' as a query parameter, e.g., by using `nc:DocumentIdentification`) then a `FilingStatusCode` can be returned for this 'filing'.

If, however, the filing status query parameter identifies something other than a single 'filing' (e.g., the query parameter provides a 'Case ID' instead of a filing-identifier) then multiple 'filings' may qualify, yet only one `ecf:FilingStatus` can be returned in the response, and only a single response can be returned for a request.

There is apparent flexibility regarding query parameters permitted for `GetFilingStatus`. Section 3.2.10 'GetFilingStatus' says "the Filing Assembly MDE MAY invoke the GetFilingStatus query operation with the filing Identifier". Although this specification statement may not be normative, it at least suggests that 'filing-identifier' is a preferred or recommended query parameter.

However, the element documentation for `statusquery:FilingStatusQueryMessage` says "this is query to get a filing's status by Filer Identification, CaseID, or Filing Number". 'Filing Number' is understood to mean the 'filing identifier'. This element documentation suggests that query parameters, other than filing-identifier, would not be disallowed by specification. When using parameters other than filing-identifier, multiple 'filings' may result.

Providing clerk review results can be further complicated when multi-episode clerk review is allowed. For example, when a submission contains a single `CoreFilingMessage` which in turn contains multiple `FilingLeadDocument` elements, and clerk review has been concluded for some, but not all lead documents, then at that moment, what is the filing status for the submission? The answer, choosing one of the four ECF provided code options, would presumably be "partially-accepted". However, if instead of accepting the documents in the first multi-episode clerk review session, the reviewed documents were rejected, then what is the filing status? ECF does not provide an "partially-rejected" filing status. If each result (e.g., RFR) for a concluded multi-episode clerk review is considered a 'filing', then the filing status in this rejection circumstance may be "rejected' and not "partially-rejected".

Now that **[ECF-v4.1]** allows multiple `CoreFilingMessage`(s) within a single `ReviewFilingRequest`, the option of multi-episode clerk review can take on a whole new dimension. See section 4.3.1.1 'Multi-Episode Clerk Review with multiple CoreFilingMessages' later in this document for additional information on this topic.

Of course, clerk review results, payment receipt information, and docketing information cannot be provided to the FAMDE from the FRMDE unless the FRMDE has access to this information. Prior to **[ECF-v4.1]**, the NotifyDocketingComplete operation was used to provide most of this information to the FRMDE (note: NotifyDocketingComplete does not provide payment receipt information to the FRMDE). Now that NotifyDocketingComplete is optional in **[ECF-v4.1]** , implementations that do not support this operation must provide some other method or methods to inform the FRMDE. These methods appear to be outside the **[ECF-v4.1]** specification.

The method for providing payment receipt information has never been fully addressed within ECF specifications. The changes in **[ECF-v4.1]** do not impact this understanding.

The `PaymentMessage` is provided to the FRMDE in the RvFR. Payment and payment receipt information is not provided to the CRMDE in the RFR. Additionally, payment information and payment receipt information are not included in the NDC. Since the `NotifyFilingReviewCompleteRequest` provides `PaymentReceiptMessage` and much or all the information provided in `PaymentReceiptMessage` originated in the `PaymentMessage` it must be presumed that a stateful protocol is envisioned.

Implementations that prefer a stateless protocol may consider extending the RFR and NDC exchanges to include payment and/or payment receipt information.

3.8.3 RecordFiling

ECF defines a RecordFiling operation on the CRMDE. As specified, the RecordFiling operation consumes one to multiple `RecordDocketingMessage` elements and one to multiple `CoreFilingMessage` elements as input parameters, returning a synchronous `MessageReceiptMessage`.

Although the **[ECF-v4.1]** illustrates the RecordFiling operation as required, through the use of bold characters in Section 3.1 'The Filing-Preparation-to-Docketing Process Model' and by not including RecordFiling within the shaded 'opt.' (i.e., optional) rectangle in Figure 4 within the same section, this section also provides the following statement:

"when the RecordFiling operation has been implemented within the same system as the ReviewFiling operation, then the RecordFiling operation need not be provided in an ECF 4.1 compliant manner."

ECF provides this flexibility to accommodate implementations in which the functions attributed to the RecordFiling operation are fulfilled by capabilities inherent to the CRMDE system. Typically, this is a Case Management System that provides e-filing clerk review capabilities. Although the **[ECF-v4.1]** "specification is not intended to define how operations must be implemented", the RecordFiling operation is understood to provide case 'docketing' functions. The details of 'docketing' are court and implementation specific.

3.9 GetFeesCalculation

[ECF-v4.1] has been modified to allow more than one `CoreFilingMessage` within a `GetFeesCalculation` request. This has been done to support newly expanded `ReviewFilingRequest` elements that may now also provide more than one `CoreFilingMessage`.

When a `ReviewFilingRequest` provides multiple `CoreFilingMessage` elements, then a single `GetFeesCalculation` request can provide all of these `CoreFilingMessage` elements within the single request and get back a single (possibly aggregate) `FeesCalculationAmount`, in the `FeesCalculationResponseMessage`, for the collection of provided `CoreFilingMessage` elements.

Alternatively, multiple `GetFeesCalculation` requests can be submitted (e.g., one for each `CoreFilingMessage`) resulting in separate `FeesCalculationAmount` elements. Since a `ReviewFilingRequest` still only provides a single `PaymentMessage`, then these separate `FeesCalculationAmount` elements may need to be summed to provide a single `PaymentMessage AllowanceCharge Amount` or may be listed individually as separate `AllowanceCharge` elements.

Although section 3.2.3 GetFeesCalculation specifies that "The Filing Assembly MDE MAY query for the fees associated with a filing by invoking the MDE's GetFeesCalculation operation, with a filing as a parameter" (i.e., singular) this should be understood as requiring at least one `CoreFilingMessage`, but also permitting multiple `CoreFilingMessage` elements as invocation parameters.

Note that in the new Section 5 'MDE Operations', the Parameters listed in Section 5.2.1 shows '`FeesCalculationQueryMessage`' as the input parameter for the GetFeesCalculation operation (and not `CoreFilingMessage`). `FeesCalculationQueryMessage` requires at least one `CoreFilingMessage` but allows multiple `CoreFilingMessage` elements.

3.10 NIEM-Core Schema Changes

The niem-core.xsd schema includes the following changes:

- `nc:ItemOtherIdentification` within `nc:ItemType` – maxOccurs changed from "1" to "unbounded".
- `nc:ObligationEntity` within `nc:ObligationType` - maxOccurs changed from "1" to "unbounded".
- `nc:OrganizationIdentification` - maxOccurs changed from "1" to "unbounded".
- `LocationCountryISO3166Alpha2Code` added – can be substituted for `nc:LocationCountry`.
- `PersonCitizenshipISO3166Alpha2Code` added – can be substituted for `nc:PersonCitizenship`.

3.11 New Country Codes Code List

The schema iso_3166.xsd has been added. This schema provides two-letter country codes.

3.12 Other Backward Compatibility Considerations

Could an **[ECF-v4.01]** MDE (e.g., FAMDE) successfully provide an **[ECF-v4.01]** ReviewFilingRequest RvFR) to an ECF v4.1 MDE (e.g., FRMDE)? This question must be considered in two parts, a) for the Core ECF specification and b) for the Web Services WSDL.

3.12.1 Backward Compatibility Considerations for ECF 4.1

If an **[ECF-v4.1]** FRMDE received an **[ECF-v4.01]** RvFR, could it be successfully processed, and could the **[ECF-v4.01]** FAMDE successfully process the synchronous **[ECF-v4.1]** ReviewFilingResponse?

Of course, to truly answer this question, testing, for each specific implementation circumstance, would be required. But generally, changes (especially XML changes) are considered to be backward compatible if they relax specifications and do not tighten specifications. For example, if an element that had been previously required is made optional, then this is a relaxation and is considered to be backward compatible. Doing the opposite would not be backward compatible.

By and large, the modifications to **[ECF-v4.1]** XML schema have relaxed criteria and have not tightened constraints. These include:

- Adding optional wrappers.xsd
- Making the upperbound cardinality of some ECF messages unbounded, such as now allowing multiple `CoreFilingMessage` elements in a `ReviewFilingRequest`.
- Revising upperbound cardinality of some elements to unbounded.
- Adding an optional Country Codes enumeration.

However, in the `CourtPolicyResponse`, a new `<RequireAsynchronousResponsesIndicator>` element has been added. This element is mandatory. But GetPolicy is not mandatory for ECF. Although implementations that do support GetPolicy may want to invoke GetPolicy prior to ReviewFiling, GetPolicy is nevertheless a separate operation. Any backward compatibility issues with GetPolicy do not affect backward compatibility for ReviewFiling (provided that Court Policy is available to FAMDE).

So, if the **[ECF-v4.01]** RvFR was composed as recommended in **[ECF-v4.01]** , i.e., consisting of a single `CoreFilingMessage` and an optional `PaymentMessage`, then theoretically, this review filing request could be understood by an **[ECF-v4.1]** ReviewFiling operation.

Namespace URI for **[ECF-v4.01]** namespaces are the same as the **[ECF-v4.1]** namespace URI, so no changes would be needed. Of course, if the **[ECF-v4.01]** implementation included implementation/court specific extensions, then court/implementation defined namespace URI may require modification (depending upon local standards and naming conventions).

Since the response XML for ReviewFiling has not been modified in **[ECF-v4.1]** , there should not be any backward compatibility issues processing the response.

3.12.2 Backward Compatibility Considerations for ECF Web Services SIP 4.1

Can an ECF v4.01 request be sent to an **[ECF-v4.1]** operation using the **[ECF-v4.0-WS-SIP-v2.01]** specification?

The short answer appears to be no. Even though wrappers.xsd is optional in **[ECF-v4.1]** , it is not clear how **[ECF-v4.1]** would operate without it. For example, the ReviewFiling operation invocation parameters specified in Section 5 of the Core Specification are `CoreFilingMessage` and (optionally) `PaymentMessage`. Although these parameters are provided in the request SOAP envelope, they are wrapped within a `ReviewFilingRequest` element which in turn is wrapped in a `ReviewFiling` SOAP Body child (e.g., exchange root) element.

So, whereas it appears that an **[ECF-v4.1]** `CoreFilingMessage` and `PaymentMessage` can be understood by an **[ECF-v4.1]** ReviewFiling operation, **[ECF-v4.0-WS-SIP-v2.01]**) could not be used to send it.

Thus, although it appears that there is a great deal of backward compatibility to **[ECF-v4.01]**, there is not 100% backward compatibility. Additionally, it appears likely that an **[ECF-v4.01]** implementation may be able to communicate with an **[ECF-v4.1]** implementation using **[ECF-v4.01]** adapted Web Services **[ECF-v4.1]** as described in Appendix D.

3.13 Backward Compatibility Summary

Since the combined **[ECF-v4.1]** and the corresponding Web Service SIP v4.1 define a complex, multi-faceted, e-filing eco-system, assessing backward compatibility is not a binary proposition. Different parts of ECF will have different backwards compatibility.

For the most part, a high degree of backward compatibility is supported. Most ECF messages created under **[ECF-v4.01]** should be successfully consumed by **[ECF-v4.1]** operations, provided the ECF messages can be delivered to the MDE operation. The exception is `GetPolicyRequest` due to the inclusion of `<RequireAsynchronousResponsesIndicator>`.

However, delivering **[ECF-v4.01]** Messages to **[ECF-v4.1]** MDE operations using Web Services is not backwardly compatible. An **[ECF-v4.01]** MDE would not be successful providing **[ECF-v4.01]** Messages to an **[ECF-v4.1]** MDE using the **[ECF-v4.0-WS-SIP-v2.01]**. However, if the ECF **[ECF-v4.01]** MDE used an **[ECF-v4.01]** adapted version of the **[ECF-WS-SIP-v4.1]** (as described in Appendix D), then this should result in successful delivery.

Nothing can be said regarding backward compatibility for court/local extensions. This must be evaluated by the implementor.

4 Limitations, Caveats and Other Items of Note

This section identifies issues, understandings and other considerations that may be important for implementations to evaluate.

4.1 Use of "Filing" Terminology

One should take note that the term "filing" is not used in a consistent manner within the **[ECF-v4.1]** (or prior) specifications. For instance,

- In Section 3.1.9 'GetFeesCalculation'), the term 'filing', in GetFeesCalculation refers to a CoreFilingMessage.
- In Section 1.4 'Terms and Definitions', the term 'filing' is described as "an electronic document …".
- In Section 3.1 'The Filing-Preparation-to-Docketing Process Model', the first paragraph ends with "other operations are optional and MAY occur within a given filing". In this context, 'filing' must be understood, not as any single document, and also not as a single `CoreFilingMessage`, since multiple `CoreFilingMessage`(s) are allowed for a single submission (e.g., `ReviewFilingRequest`), but as a complete submission (that may contain multiple `CoreFilingMessage`(s) (for one or many cases) and multiple documents (both lead and connected)).
- In Section 3.2.4 'ReviewFiling' says "the Filing Assembly MDE MUST submit the filing to the court by invoking the ReviewFiling operation". In this context, 'filing' appears to mean '`ReviewFilingRequest`'. As of **[ECF-v4.1]** , a `ReviewFilingRequest` may contain multiple `CoreFilingMessage`(s).
- The `RecordDocketingCallbackMessage` (RDCM) includes the `ecf:FilingStatus` element. In **[ECF-v4.01]** , a RDCM only allows one `ReviewedLeadDocument` (and zero to many child `ReviewedConnectedDocument` elements). Although the cardinality for RDCM within NDC has been modified in **[ECF-v4.1]** to permit unlimited `ReviewedLeadDocument`(s) within a single RDCM, the prior understanding for 'filing' still lingers, i.e., even in **[ECF-v4.1]** , there remains a single `FilingStatus` element within a RDCM. Since there is only a single lead document in a RDCM, the inclusion of a single `FilingStatus` infers that 'filing' means 'lead document and its connected documents'. As such, when an RvFR provides a `CoreFilingMessage` with multiple Lead Documents, from the RDCM `FilingStatus` perspective, this is multiple 'filings'.
- In Section 3.2.5 'ServeFiling', it states "the Filing Assembly MDE MAY serve the entire filing, to other parties in the case by invoking the ServeFiling operation on the ServiceMDE". In this context, the term 'filing' should be understood as a single `CoreFilingMessage` (including the documents contained or referenced therein) from within a `ReviewFilingRequest`.

Overall, when the term 'filing' appears within the specification, it often refers to either a single `CoreFilingMessage` (that may provide one or many documents, lead and connected) and sometimes refers to a complete `ReviewFilingRequest` (that may provide many `CoreFilingMessage` elements). In at least one instance (e.g., `FilingStatus`) 'filing' refers to a single lead document and its connected documents. The term 'filing' rarely means 'an electronic document' as stated in the 'Terms and Definitions' section.

Consider `FilingStatusResponseMessage`. This message is restricted to a single `nc:Case` (or case type specific element via substitution), a single `nc:DocumentIdentification`, and a single `ecf:FilingStatus`. This structure does not lend itself to supporting multiple `CoreFilingMessage`(s)

within a single ‘filing’ (e.g., ’`ReviewFilingRequest`’), especially when multiple cases are provided within a single ‘filing’.

4.2 Filing Identifier

Also consider “filing identifier” (e.g., Section 3.3.1.5). It seems reasonable that single (and locally (e.g., court) unique) “filing identifier” is assigned to a single ‘filing’.

The specification language provided in Section 3.3.1.5 ‘Filing Identifiers’ suggests that a single ‘filing identifier’ is assigned for a single `ReviewFilingRequest` (e.g., “will be generated by the court in response to a ReviewFiling operation”). This single ‘filing identifier’ to a single `ReviewFilingRequest` is also implied within Section 3.2.4 ‘ReviewFiling’, as: “The Filing Review MDE responds synchronously with a receipt message that includes the filing identifier issued by the court.” (singular).

More practical is that a unique ‘filing identifier’ would be assigned for each `CoreFilingMessage`. As such, when a single `ReviewFilingRequest` provides multiple `CoreFilingMessage`(s), then multiple ‘filing identifiers’ should be assigned (i.e., one for each `CoreFilingMessage`). In this context ‘filing’ (as designated by a ‘filing identifier’) refers to a `CoreFilingMessage`.

The Core specification does not specify how ‘filing identifiers’ are specified (e.g., which element(s) to use, etc.), for contrast see Section 3.3.1.8 ‘Filer and Party Identifiers’ in which a non-normative example is provided).

The Core specification does not establish where, when or how filing identifiers are generated. The implication from Section 3.2.4, quoted above, is that assignment, and perhaps generation, of filing identifiers is done by the FRMDE. Nevertheless, it does seem clear that the filing identifiers are expected to be provided to the FAMDE in the `ReviewFilingResponse` within `MessageReceiptMessage`. Since this response is synchronous, then filing identifiers must either be generated in the FRMDE or pre-generated then assigned by the FRMDE, and all filing identifiers assigned for a single `ReviewFilingRequest` must be returned together.

4.3 Multiple CoreFilingMessages in ReviewFilingRequest

The `ReviewFilingRequest` may now include more than one `CoreFilingMessage`. There are few restrictions imposed on this capability. Implementors are cautioned about exploiting this feature.

Although not specifically called out in the Core specification, one limitation is that all `CoreFilingMessage`(s) should be destined to the same court, even though the `ReviewFilingRequest` structures and schema would not prevent multiple courts. This single court destination is implied by the Core specification when it states (Section 3.2.4 ‘ReviewFiling’) “The Filing Assembly MDE MUST submit the filing to the court”. In this context, ‘filing’ would refer to a single `ReviewFilingRequest`. The use of the term “the court” implies a single court.

Mixed cases within a single `ReviewFilingRequest` are also not precluded, provided all the cases are at the same court. Since only a single `PaymentMessage` is allowed within the `ReviewFilingRequest` for the multiple mixed-case `CoreFilingMessage`(s), this may provide natural governance limiting the mix of cases.

Multiple filing-identifiers are recommended, one for each `CoreFilingMessage`. Filing-identifiers are to be provided to the FAMDE in the ReviewFiling response. The Core specification does not identify how this would be done within a single ReviewFiling response. In fact, the Core specification does not even identify how a single filing-identifier should be returned in a `MessageReceiptMessage` to the FAMDE from the FRMDE.

The Core specification does not preclude rejection of submissions prior to clerk review (e.g., by an EFM). Rejection could be due to not well-formed XML or XML not valid to schema, or other cause, such as invalid values, documents exceeding size limits, corrupt documents, viruses, etc.

When a submission provides a single `CoreFilingMessage`, then communicating this rejection status is simpler since the `MessageReceiptMessage` returned is addressing a single `CoreFilingMessage`.

However, when a `ReviewFilingRequest` provides more than one `CoreFilingMessage`, then the use of a single `MessageReceiptMessage` is more challenging. Although `MessageReceiptMessage` permits multiple errors to be reported, ECF does not provide any mechanism for associating specific errors to specific `CoreFilingMessage`(s). At present, ECF provides no guidance for this circumstance.

Additionally, the Core specification does not preclude implementations from splitting up `ReviewFilingRequest`(s) following submission by the FAMDE. For example, a single `ReviewFilingRequest` that contains multiple `CoreFilingMessage`(s) could be divided into multiple exchange units by an EFM (e.g., one exchange for each `CoreFilingMessage`), before being provided to the FRMDE (e.g., clerk review). In this circumstance, multiple 'filing identifiers' would be especially useful.

When an EFM rejects one or more, but not all `CoreFilingMessage`(s) within a single `ReviewFilingRequest`, and then sends the un-rejected `CoreFilingMessage`(s) on to the FRMDE, then any `MessageReceiptMessage` returned to the FAMDE should be clear and specific when identifying the rejected `CoreFilingMessage`(s). ECF does not provide any guidance on this within the Core specification.

One approach that implementations may consider, would be to leverage the `nc:DocumentIdentification` element within `MessageReceiptMessage` (MRM), using this element to list all `CoreFilingMessage`(s) to which the MRM applies (e.g., by providing the `CoreFilingMessage`'s filing identifier in `nc:DocumentIdentification/nc:IdentificationID`, presuming that one filing-identifier is assigned to each `CoreFilingMessage`). So, if the `ReviewFilingRequest` provides 8 `CoreFilingMessage`(s), and each `CoreFilingMessage` is provided a unique filing-identifier, then the response `MessageReceiptMessage` would include 8 instances of `nc:DocumentIdentification`, with each instance returning a unique filing-identifier.

By itself, the above proposal still does not allow a specific `ecf:Error` to be associated with a specific `CoreFilingMessage`. This linkage may perhaps be provided by utilizing attributes (e.g., `s:id`, `s:metadata`, and/or `s:linkMetadata`).

When a `ReviewFilingRequest` provides multiple `CoreFilingMessage`(s), there are implications for clerk review (e.g., multi-episode clerk review, etc.) and for `RecordFilingRequest`(s).

4.3.1.1 Multi-Episode Clerk Review with multiple CoreFilingMessages

Multi-Episode Clerk Review occurs when a full `ReviewFilingRequest` is reviewed in more than one clerk review session and when submission documents and/or `CoreFilingMessage`(s) are accepted or rejected (or other disposition) in each session without accepting or rejecting all submission documents within a single session, and when the intermediate results are forwarded to the next operation (i.e., RecordFiling). As such, it takes two or more sessions to provide clerk review dispositions for all documents within a submission. These sessions may be minutes, hours or even days apart.

Multi-Episode clerk review is possible even when a `ReviewFilingRequest` contains a single `CoreFilingMessage`. It may have greater probability when the `ReviewFilingRequest` contains multiple `CoreFilingMessage`(s).

When a `ReviewFilingRequest` contains a single `CoreFilingMessage`, multi-episode clerk review is possible when the `CoreFilingMessage` provides multiple filing documents, especially multiple `FilingLeadDocument`(s). Generally, when a `FilingLeadDocument` is either accepted or rejected all of its `FilingConnectedDocument`(s) are also either accepted or rejected (e.g., a `FilingLeadDocument` and its `FilingConnectedDocument`(s) are generally reviewed as a unit).

When multi-episode clerk review occurs, then as a result, there will be multiple RecordFiling operation requests for a single `ReviewFilingRequest` (when the ECF RecordFiling operation is used).

4.3.1.2 RecordFilingRequests for multiple CoreFilingMessages

Even when a submission (e.g., `ReviewFilingRequest`) contains a single `CoreFilingMessage`, there may be multiple `RecordDocketingCallbackMessage`(s). In a fully reviewed `ReviewFilingRequest`, there would be one `RecordDocketingCallbackMessage` for each `FilingLeadDocument` in **[ECF-v4.01]**, but now with **[ECF-v4.1]**, multiple ReviewedLeadDocument(s) may be accommodated within a single `RecordDocketingCallbackMessage`.

When the `ReviewFilingRequest` contains multiple `CoreFilingMessage`(s) then there are several possible permutations. The first, and simplest permutation would be to provide a single `RecordFilingRequest` to the RecordFiling operation that contained all of the `ReviewFilingRequest`'s provided `CoreFilingMessage`(s), and also contained one, and only one, corresponding `RecordDocketingCallbackMessage` for each `CoreFilingMessage`. For this simplest, most straight forward option, each `RecordDocketingCallbackMessage` would contain a `ReviewedLeadDocument` element for each `FilingLeadDocument` in its corresponding `CoreFilingMessage`.

4.4 Service List Registry

Court service list registry – Section 3.2.2 'GetServiceInformation' identifies a "Court's registry" as the source for case participant service information (e.g., contact information). The concept of "court registry" is not defined in the specification. As such, Core specification mandates, such as "there MUST be only one such registry per court" and "the Court Record MDE MUST have access to the court's registry", are unclear (and perhaps also unenforceable, as in not required for a complete, or compliant implementation).

Also, keep in mind that GetServiceInformation is an optional operation that implementations need not support.

4.5 Hub Service

Hub Service – this type of service implementation is raised in Section 3.2.5 'ServeFiling'. 'Hub Service' also appears in Section 3.2.2 'GetServiceInformation'.

A 'Hub Service MDE' is defined in Section 1.4 'Terms and Conditions' as "A centralized Service MDE capable of receiving a single set of service notifications for all parties registered for electronic service in a case and transmitting the service notifications to the Service MDEs registered to each party in the case."

A Hub Service is not a true MDE and instead is a proxy for one or more other MDE implementations. The Core specification only considers 'Hub Services' for the Service MDE, specifically ServeFiling and GetServiceInformation.

Other possible uses for 'Hub Services' could be considered. For example, a centralized Electronic Filing Manager (EFM) could service multiple FRMDEs, e.g., one or more in one or many courts. This same Hub Service could also handle all GetPolicy requests for all supported CRMDEs.

4.6 Implementation Namespace

[ECF-WS-SIP-v4.1] provides 4 example implementation WSDL in addition to the four (4) specification provided, normative base WSDL, e.g., one for each MDE. Implementation WSDL are expected to import the specification provided MDE base WSDL. As shown in the provided example implementation WSDL, each implementation WSDL is expected to define an implementation specific namespace.

The ECF specifications do not provide guidelines for these namespace names. The implementation examples all provide the same namespace name:

```
urn:oasis:names:tc:legalxml-courtfiling:schema:wsdl:WebServices-ImplementationExample-4.1
```

Although implementations MAY to continue the **[ECF-v4.01]** practice of providing a single implementation specific namespace, they SHOULD define up to four implementation specific namespaces, one namespace for each MDE.

4.7 Payment Maximum Amount

In Section 3.3.3.2 'PaymentMessage', the specification states "The payment MAY include a maximum amount for the payment if some latitude is needed to accomplish the filing." The specification does not identify the element or elements used for this purpose (e.g., ecf:MaximumAmount). There is no obvious element (based on element names and descriptions) for this maximum amount.

Perhaps the specification statement is intended to suggest that courts/implementations are free to impose the largest allowed fee(s).

Additionally, there does not appear to be any support in GetFeesCalculation, either in the request or in the response, for a payment maximum amount.

The element cbc:PaidAmount MAY be used to indicate a maximum amount that MAY be charged for the submission rather than an amount actually paid. Courts SHOULD consider using Human Readable Court Policy to clarify how cbc:PaidAmount is used.

4.8 Differences with ECF 5.x

Although many of the changes that appear in **[ECF-v4.1]** are intended to better align with **[ECF 5.01]** some important differences remain.

4.8.1 No new operations

No new operations, optional or otherwise, have been included in **[ECF-v4.1]**. ECF 5.x introduces several new operations, i.e., CancelFiling, DocumentStampInformation, GetCourtSchedule, RequestCourtDate, ReserveCourtDate, AllocateCourtDate, and NotifyCourtDate.

4.8.2 Case-type-specific elements

Case type specific elements in **[ECF-v4.1]** are still only intended for use for case initiation filing submissions and are not intended for use with subsequent filing submissions.

This distinction is most noticeable in that with ECF 5.x the `nc:Case` element may include `ecf:CaseAugmentation` and `j:CaseAugmentation`, whereas with ECF 4.x, `nc:Case` does not provide any augmentation elements.

The ECF TC understands that some or many ECF 4.x implementations may also use case type specific elements within subsequent filing submissions.

4.9 NIEM Version

The version of **[NIEM]** has not been modified for **[ECF-v4.1].** Updating the NIEM version from 2.0 to 2.1 was considered for **[ECF-v4.1]**. Minor version number updates in NIEM are supposed to be backward compatible. However, when investigating this, it was discovered that the element `j:StatuteOffenseIdentification` had been removed from `j:StatuteType` in NIEM 2.1.To preserve the greatest degree of backward compatibility, it was decided to stay with NIEM 2.0 for **[ECF-v4.1]** .

Since the same version of NIEM is used, this means that any code-lists used by ECF from NIEM will have the very same code list values as available in **[ECF-v4.01]** . This may have implications on newer implementations. For example, if new countries have been added or former countries have vanished or country names have been changed, then the older NIEM code lists (e.g., ISO 3166) will not contain these new or updated values and will continue to include obsolete values. To mitigate this limitation, future versions, such as **[ECF 5.01],** are expected to rely more heavily on Genericode (gc) code lists and less so on schema enumerated code lists.

4.10 ECF Conformance and Compliance

[ECF-v4.1] does not provide OASIS recommended conformance clauses, relying instead on a blanket statement in section 8 ‘Conformance’ in the Core specification:

“*An implementation conforms with the Electronic Court Filing Version 4.1 if the implementation meets the requirements in Sections 1-6 including conformance with the XSD schemas and* **[Genericode]** *code lists* referenced in Section 3 and 4”

The TC may consider writing specific numbered conformance clauses in the future, as required by OASIS (see docs.oasis-open.org/templates/TCHandbook/ConformanceGuidelines.html).

In addition to the blanket conformance statement provided in the Core specification, the following additional compliance statements are included in **[ECF-v4.1]**:

- Section 2.2 ‘Major Design Elements’ of the Core specification provides: “*An ECF 4.1-compliant implementation may implement one or more of the MDEs defined in the specification but a complete ECF 4.1 system MUST include at least one each of the Filing Assembly, Filing Review and Court Record MDEs.*” The words “*one each*” imply that, for any given MDE, there may be multiple components that provide the operations implemented for that MDE. These words should not be understood as multiple MDEs, such as multiple FAMDEs within an implementation. Instead, there would be a single FAMDE, but there may be multiple FAMDE service providers. The Section 2.2 statement cited above is further illustrated with “*For instance, a court may decide to provide certain MDEs and allow private providers to furnish the remaining MDE*s”.
- This section also provides” In order to be compliant with ECF 4.1, an MDE MUST support all required operations for that MDE. However, in an ECF 4.1 system that does not support electronic service, the operations associated with the Legal Service MDE are not required”.
- Section 3.1 ‘The Filing-Preparation-to-Docketing Process Model’ provides “*The ReviewFiling and RecordFiling operations are required in a complete ECF 4.1 system as prescribed in Section 2.2.”* However, when the RecordFiling operation has been implemented within the same system as the ReviewFiling operation, then the RecordFiling operation need not be provided in an ECF 4.1 compliant manner”.
- Section 2.2 ‘Core vs. Profiles’ provides “*In order to be compliant, an implementation of the ECF specification MUST implement the core specification and at least one service interaction profile and one document signature profile*”.
- Section 2.4.1 ‘Human-Readable Court Policy’ states “*To be compliant with the ECF 4.1 specification, each court MUST publish a human-readable court policy that MUST include each of the following*” and then provides a listing of information that must be included.

In the above, three terms are used: 1) conformant, 2) compliant, and 3) complete.

It appears that an implementation can be ‘complete’ and not also be either ‘conformant’ or ‘compliant’.

To be ‘complete’, an implementation must:

- Provide the ReviewFiling operation, presumably in an **[ECF-v4.1]** compliant manner, and
- Provide the RecordFiling operation, however the RecordFiling operation need not be **[ECF-v4.1]** compliant when RecordFiling and ReviewFiling are within the same system, and
- Include the Filing Assembly MDE, the Filing Review MDE and the Court Record MDE.

To be ‘compliant’, an implementation must:

- Implement a Service Interaction Profile (e.g., **[ECF-WS-SIP-v4.1]**), and
- Implement at least one document signature profile, and
- Publish human-readable court policy as required in section 2.4.1 of the Core specification, and
- Implement at least one MDE, but may implement more than one MDE, and
- Provide all required operations for implemented MDEs.

To be ‘conformant’, an implementation must:

- Be a ‘compliant’ implementation, and
- Meet the requirements stated in sections 1 – 6 in the Core specification, and
- Conform with **[ECF-v4.1]** schema, and
- Conform with Genericode lists provided in sections 3 and 4 in the Core specification.

It is not clear what it means to conform with the Genericode lists. Section 3.3.2 ‘Code Lists’ provides a listing of normative code lists. Some of the included code lists are Genericode lists. **[ECF-v4.1]** provides 16 Genericode lists in the ‘gc’ folder. Each of these Genericode lists contain lists of codes.

It is not clear whether Genericode compliance requires the use of all 16 Genericode lists (when applicable) as provided by **[ECF-v4.1]** or whether the code lists can be extended or contracted.

The statement “when applicable” as used above, means that the Genericode list is used when the exchange and/or case type includes elements controlled by the code list. Determining this applicability, is in itself, tenuous. There are no normative statements in the **[ECF-v4.1]** specification that associate specific elements with specific Genericode lists, as is done with schema enumerated code lists. It appears that the association is ‘suggested’ by Genericode filename, e.g., the Genericode file ‘ECF-4.1-FilingStatusCode.gc’ governs the contents of the ‘`FilingStatusCode`’ element (fortunately this element name only appears in one namespace). Some ECF elements provide non-normative documentation in the element definition, provided in schema, that may aid in this association. For example, the element description for `ecf:FilingStatusCode` is “Status of the filing as determined by the system sending the callback. Allowable values defined in ECF-4.0-FilingStatusCode.gc.” Clearly the reference should be to ‘ECF-4.1-FilingStatusCode.gc’ rather than to ECF-4.0.

To aid in the determination whether all requirements stated in Core specification sections 1 - 6 have been met, and in the absence of conformance clauses, a listing of specification normative statements is included in Appendix F.

Appendix A. Informative References

This appendix contains the references that are used in this document.

While any hyperlinks included in this appendix were valid at the time of publication, OASIS cannot guarantee their long-term validity.

[ECF-v4.01]

Electronic Court Filing Version 4.01. Edited by Adam Angione and James Cabral. Latest stage: http://docs.oasis-open.org/legalxml-courtfiling/specs/ecf/v4.01/ecf-v4.01-spec/ecf-v4.01-spec.html.

[ECF-v4.01-errata02]

Electronic Court Filing Version 4.01 Errata 02. Edited by James Cabral and Gary Graham. 07 July 2015. OASIS Approved Errata. http://docs.oasis-open.org/legalxml-courtfiling/specs/ecf/v4.01/ecf-v4.01-spec/errata02/os/ecf-v4.01-spec-errata02-os.html. Latest version: http://docs.oasis-open.org/legalxml-courtfiling/specs/ecf/v4.01/ecf-v4.01-spec/errata02/ecf-v4.01-spec-errata02.html.

[ECF-v4.0-WS-SIP-v2.01]

Electronic Court Filing 4.0 Web Services Service Interaction Profile Version 2.01. Edited by Adam Angione. 09 August 2011. http://docs.oasis-open.org/legalxml-courtfiling/specs/ecf/v4.0/ecf-v4.0-webservices-spec/v2.01/ecf-v4.0-webservices-spec-v2.01.html

[ECF-v4.1]

Electronic Court Filing Version 4.1. Edited by James Cabral, Gary Graham, and Philip Baughman. Latest stage: https://docs.oasis-open.org/legalxml-courtfiling/ecf/v4.1/ecf-v4.1.html.

[ECF-WS-SIP-v4.1]

Electronic Court Filing Web Services Service Interaction Profile Version 4.1. Edited by James Cabral, Gary Graham, and Philip Baughman. 07 December 2022. OASIS Committee Specification Draft 01. https://docs.oasis-open.org/legalxml-courtfiling/ecf-webservices/v4.1/csd01/ecf-webservices-v4.1-csd01.html. Latest stage: https://docs.oasis-open.org/legalxml-courtfiling/ecf-webservices/v4.1/ecf-webservices-v4.1.html.

[ECF-v5.01]

Electronic Court Filing Version 5.01. Edited by James Cabral, Gary Graham, and Philip Baughman. 06 December 2022. OASIS Committee Specification Draft 02. https://docs.oasis-open.org/legalxml-courtfiling/ecf/v5.01/csd02/ecf-v5.01-csd02.html. Latest stage: https://docs.oasis-open.org/legalxml-courtfiling/ecf/v5.01/ecf-v5.01.html.

[Interoperability Guidelines]

Jacques Durand, https://www.oasis-open.org/policies-guidelines/interoperability-guidelines/

[NIEM]

https://niemopen.org/

[Portable Media Messaging Profile]

Portable Media Messaging Profile 1.0, Edited by Roger Winters, November 15, 2005. OASIS Committee Draft. https://docs.oasis-open.org/legalxml-courtfiling/specs/ecf/v3.0/ecf-v3.0-portablemedia-spec/ecf-v3.0-portablemedia-spec-cd01.doc

Appendix B. Acknowledgments

The following individuals have participated in the creation of this committee note and are gratefully acknowledged:

Participants:

Gary Graham, Arizona Supreme Court

James Cabral. InfoTrack US

B.1 Special Thanks

Special thanks from the technical committee go to Gary Graham for leading the development of this committee note.

B.2 Participants

The following individuals were voting members of this Technical Committee during the creation of this document and their contributions are gratefully acknowledged:

Philip Baughman, Tyler Technologies, Inc.

James Cabral, InfoTrack US

Eric Eastman, InfoTrack US

Gary Graham, Arizona Supreme Court

Ryan Foley, ImageSoft, Inc.

George Knecht, InfoTrack US

Mark Leong, Arizona Supreme Court

James, McMillan, National Center for State Courts

Enrique Othon, Tyler Technologies, Inc.

Jim Price, Arizona Supreme Court

Brock Rogers, File & ServeXpress

Patrick Wallace, Tyler Technologies, Inc.

Appendix C. Wrappers.xsd structures

The table below lists all type-structures defined within wrappers.xsd. For each type-structure, the element derived from this structure is provided in the middle column, i.e., “Derived Element”. Often this derived element is the operation invocation parameter, named as required in the **[ECF-WS-SIP-v4.1]** (e.g., Section 2.5 ‘Request and Operation Invocation’; named using the operation name). The rightmost column provides the name of the invocation parameter. Request rows with ‘N/A’ (not applicable) in the rightmost column identify request Type Structures that are used to derive elements that in turn are used to derive additional Type Structures. The rightmost column is grey for response types, since responses are never used as invocation parameters.

To illustrate, the element ‘`NotifyFilingReviewCompleteRequest`’ (derived from ‘`NotifyFilingReviewCompleteRequestType`’) is the exclusive content of ‘`NotifyFilingReviewCompleteType`’ (from which ‘`NotifyFilingReviewComplete`’ has been derived). When the NotifyFilingReviewComplete operation is invoked, it uses the `NotifyFilingReviewComplete` element as the invocation parameter.

Type Structure	Derived Element	WSSIP Operation Invocation
GetPolicyRequestType	GetPolicy	GetPolicy
GetCaseListRequestType	GetCaseList	GetCaseList
GetCaseRequestType	GetCase	GetCase
GetDocumentType	GetDocument	GetDocument
GetFeesCalculatonRequestType	GetFeesCalculation	GetFeesCalculation
GetFilingListRequestType	GetFilingList	GetFilingList
GetFilingStatusRequestType	GetFilingStatus	GetFilingStatus
GetServiceInformationRequestType	GetServiceInformation	GetServiceInformation
ServeFilingRequestType	ServeFiling	ServeFiling
NotifyDocketingCompleteRequestType	NotifyDocketingComplete	NotifyDocketingComplete
NotifyFilingReviewCompleteRequestType	*NotifyFilingReviewCompleteRequest*	N/A
NotifyFilingReviewCompleteType	NotifyFilingReviewComplete	NotifyFilingReviewComplete
RecordFilingRequestType	*RecordFilingRequest*	N/A
RecordFilingType	RecordFiling	RecordFiling
ReviewFilingRequestType	*ReviewFilingRequest*	N/A
ReviewFilingType	ReviewFiling	ReviewFiling
GetPolicyResponseType	GetPolicyResponse	
GetCaseListResponseType	GetCaseListResponse	

GetCaseResponseType	GetCaseResponse	
GetDocumentResponseType	GetDocumentResponse	
GetFeesCalculationResponseType	GetFeesCalculationResponse	
GetFilingListResponseType	GetFilingListResponse	
GetFilingStatusResponseType	GetFilingStatusResponse	
GetServiceInformationResponseType	GetServiceInformationResponse	
ServeFilingResponseType	ServeFilingResponse	
NotifyDocketingCompleteResponseType	NotifyDocketingCompleteResponse	
NotifyFilingReviewCompleteResponseType	NotifyFilingReviewCompleteResponse	
RecordFilingResponseType	RecordFilingResponse	
ReviewFilingResponseType	ReviewFilingResponse	

In the table above, names in **bold** are request type structures that do not include ‘Request’ within the type name. Names highlighted in *italics* are type structures and elements that do not result in an invocation parameter named element.

Appendix D. Adapting ECF v4.1 Web Services SIP to ECF 4.01

D.1 Summary

This section explores the use of the **[ECF-WS-SIP-v4.1]** with **[ECF-v4.01]**. The approach considered here will be to use the **[ECF-WS-SIP-v4.1]** four MDE specific WSDL, minimally modified as necessary, to work with **[ECF-v4.01]**.. To do this, the new **[ECF-v4.01]**.wrappers.xsd will need to be used with **[ECF-v4.01]**.. Wrappers.xsd will require minimal modification for use in **[ECF-v4.01]**.

D.2 Introduction

As originally conceived, SIP specifications are independent of the main ECF specification, permitting implementers to choose one or more SIPs from a library of available SIPs. This library has always been small, consisting of the **[Portable Media Messaging Profile],** Web Services SIP and the committee draft of an IBM MQ SIP. There have been recent conversations about other possible SIPS such as a REST/XML SIP (see ECF TC F2F Meeting Minutes for 12-07-2022), and once an Email SIP had been considered.

Although ECF implementers may choose from a library of SIPs (one or multiple) for an implementation, mixing and matching has not been intended to operate across different versions.

For example, the Introduction for **[ECF-v4.0-WS-SIP-v2.01]** states that it is “for use with ECF 4.0 specification” and also provides section 1.1 “Relationship to ECF 4.0 Specifications”. The reference to ECF 4.0 is understood as ECF v4.0 or its minor variants, such as **[ECF-v4.01]**, and not to the whole family of ECF 4.x specifications (including **[ECF-v4.1]**).

Similarly, the Introduction in **[ECF-WS-SIP-v4.1]** states, “for use with the **[ECF-v4.1]** specification” and also provides section 1.1 “Relationship to ECF 4.1 Specifications”. This specification also states that the new specification replaces or supersedes the **[ECF-v4.0-WS-SIP-v2.01]**Specification.

D.3 Rationale

Can the Web Services SIP 4.1 be used successfully with ECF 4.01?

First off, the **[ECF-WS-SIP-v4.1]** is much cleaner and more practical than the older v2.0 and v2.1 SIPs, so the above question is not just academic, it also has practical value.

The existing Web Services SIPs for **[ECF-v4.1]** have difficulties long considered by the ECF TC. These include:

- A gap exists between the **[ECF-v4.1]** and its two committee recommended Web Services SIPs (i.e., v2.0 and v2.1). This gap must be filled by implementations through the use of implementation specific exchange schema. When using version adapted wrappers.xsd and WS SIP v4.1, no exchange schema gaps will exist.
- The WS SIPs for **[ECF-v4.1]** specify, in Section 2.5 “Request and Operation invocation”, “Each message transmission MUST identify the operation being invoked within the SOAP Body only; the (qualified) operation name MUST be the qualified name of the first child element of the SOAP body element, as called for in section 7.1 of the [SOAP 1.1] specification.” Whereas **[ECF-WS-**

SIP-v4.1] conforms to the requirement within Section 2.5, older versions of Web Services SIP do not.

- The issue described above was raised during the ICJIS Springboard project (see https://www.oasis-open.org/apps/org/workgroup/legalxml-courtfiling/download.php/54588/ECF%20Springboard%20Quality%20Assurance%20Review%20v.1.0.1.docx) and was never resolved.
- Given the issue identified above, many ECF 4 implementers have followed an implementation implied by the ECF Web Services SIP provided WSDL and therefore may not have provided Section 2.5 conformant implementation specific exchange schema. This results in SOAP Body contained XML that is not conformant the WSDL xsd and relevant schema.

A non-normative conformant example (D.1) is illustrated in Appendix D of the WS SIP specification document. This example is copied below, with **bold** added:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=boundary;
   type=”application/xop+xml”;
   start="Envelope"
   start-info=”text/xml”

--boundary
Content-Type:application/xop+xml;
 text/xml; charset="UTF-8"
Content-Transfer-Encoding: 8bit
Content-ID: Envelope

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
    <env:Body xmlns:types="http://example.com/some-namespace">
        <types:ReviewFiling>

          <CoreFilingMessage>
                …
          </CoreFilingMessage>

          <PaymentMessage>
                …
          </PaymentMessage>

        </types:ReviewFiling>
    </env:Body>
</env:Envelope>
```

The example shows compliance with section 2.5 by providing <types:ReviewFiling> as the immediate child element beneath the SOAP <env:Body> element. However, in terms of ‘wsdl alignment’, this is not the resultant SOAP when a SOAP request is generated from specification provided WSDL. This can be verified using either XMLSpy or SoapUI.

D.4 Considerations

The language in the **[ECF-WS-SIP-v4.1]** appears to require the use of ECF-4.0-WebServicesProfile-Definitions.wsdl, however nothing within the specification suggests that modifications to this wsdl are prohibited. Even if properly interpreted, since **[ECF-WS-SIP-v4.1]** will be used instead of **[ECF-v4.0-WS-SIP-v2.01]** ECF-4.0-WebServicesProfile-Definitions.wsdl will not be used.

When an **[ECF-v4.01]**, MDE sends an exchange (e.g., **[ECF-v4.01]**, RecordFilingRequest) to an ECF v4.1 MDE, using an **[ECF-v4.01]** adaptation of the **[ECF-WS-SIP-v4.1]**, as described in this Appendix,

then Service Interaction Profile Identifier, provided as SendingMDEProfileCode, should be the WS-SIP v4.1 identifier, e.g., `urn:oasis:names:tc:legalxml-courtfiling:schema:xsd:WebServices-4.1`

[ECF-v4.1] provides the new wrappers.xsd. This wrappers.xsd is required for use with Web Services 4.1 and will need to be incorporated into **[ECF-v4.01]**. Modifications will need to be made to wrappers.xsd to accommodate **[ECF-v4.01]**. Depending upon XML environments and folders, the **[ECF-v4.01]** adaptation of wrappers.xsd may not need to be named using a different filename.

By comparison, **[ECF-WS-SIP-v4.1]** is divided into 4 separate WSDLs, one for each MDE. Prior Web Services SIPs provided a single WSDL for all MDEs. Depending upon XML environments, WSDL filenames may be the same as the **[ECF-v4.1]** source files, or may need to be named differently.

D.5 Approach

1. Copy wrappers.xsd from **[ECF-v4.1]**.and modify for **[ECF-v4.01]**.
2. Copy 4 MDE specific WSDL from **[ECF-WS-SIP-v4.1]** and modify for **[ECF-v4.01]**.
3. Copy/create implementation MDE specific WSDL.

D.6 Wrappers.xsd modifications for ECF v4.01:

1. In <schema> element, modify all namespace URI for ECF namespaces, replacing ‘4.1’ with ‘4.0’.

For example,

```
xmlns:docketcb="urn:oasis:names:tc:legalxml-courtfiling:schema:xsd:RecordDocketingCallbackMessage-4.1"
```

Must be revised, replacing ‘4.1’ with ‘4.0’

2. Modify the targetNamespace URI, replacing ‘4.1’ with ‘4.0’.

For example,

```
targetNamespace="urn:oasis:names:tc:legalxml-courtfiling:schema:xsd:MessageWrappers-4.1"
```

Must be revised, replacing ‘4.1’ with ‘4.0’

3. Modify the value for the version attribute, setting it to “4.0” instead of “4.1”.

For example,

```
version="4.1"
```

Must be revised, replacing ‘4.1’ with ‘4.0’

4. Within ‘imports’, modify both namespace URI and schemaLocation, replacing ‘4.1’ with ‘4.0’.

For example:

```
<xsd:import namespace="urn:oasis:names:tc:legalxml-courtfiling:schema:xsd:ReviewFilingCallbackMessage-
4.1" schemaLocation="message/ECF-4.1-ReviewFilingCallbackMessage.xsd"/>
```

Must be revised, replacing ‘4.1’ with ‘4.0’

D.7 WSDL modifications for ECF 4.01:

In each of the four MDE specific WSDL:

1. Attribute values within the <definitions> element require modification to replace the version number ‘4.1’ within namespace URI with ‘4.0’. This applies to the ‘targetNamespace’ attribute, the ‘xmlns:tns’ attribute and the ‘xmlns:wrappers’ attribute.

   For example:

```
<definitions targetNamespace="urn:oasis:names:tc:legalxml-
courtfiling:schema:wsdl:FilingAssemblyMDE-4.1"
     xmlns:tns="urn:oasis:names:tc:legalxml-courtfiling:schema:wsdl:FilingAssemblyMDE-4.1"
     xmlns:wrappers="urn:oasis:names:tc:legalxml-courtfiling:schema:xsd:MessageWrappers-4.1"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/"
     xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
     xmlns:wsrmp="http://docs.oasis-open.org/ws-rx/wsrmp/200702"
     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
```

   Must be revised, replacing ‘4.1’ with ‘4.0’

2. Within soapAction for each operation (e.g., “NotifyFilingReviewComplete”), replace ‘4.1’ with ‘4.0’.

   For example:

```
<soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="NotifyFilingReviewComplete">
    <soap:operation
            soapAction="urn:oasis:names:tc:legalxml-courtfilingschema:wsdl:FilingAssemblyMDE-
4.1\NotifyFilingReviewComplete"/>
    <input>
            <soap:body use="literal"/>
    </input>
```

   Must be revised, replacing ‘4.1’ with ‘4.0’

In addition to the four MDE specific WSDL provided in WS-SIP v4.1, there are 4 implementation examples WSDL also provided. The import element in implementation MDE WSDL will also need to be modified to replace “4.1” with “4.0”:

```
<import namespace="urn:oasis:names:tc:legalxml-courtfiling:schema:wsdl:FilingAssemblyMDE-4.1"
location="../FilingAssemblyMDE.wsdl"/>
```

Must be revised, replacing ‘4.1’ with ‘4.0’

D.8 Examples

The following ECF 4.01 adapted files are provided in the “ecf-4.1-wssip-adapted-to-4.01” folder:

- wrappers.xsd
- wsdl/CourtRecordMDE.wsdl
- wsdl/FilingAssemblyMDW.wsdl
- wsdl/FilingReviewMDE.wsdl
- wsdl/ServiceMDE.wsdl
- wsdl/examples/CourtRecordMDE-ImplementationExample.wsdl
- wsdl/examples/FilingAssemblyMDE-ImplementationExample.wsdl
- wsdl/examples/FilingReviewMDE-ImplementationExample.wsdl
- wsdl/examples/ServiceMDE-ImplementationExample.wsdl

Appendix E. Revision History

Revisions made since the initial stage of this numbered Version of this document are tracked here.

Revision	Date	Editor	Changes Made
0.1	2023-08-17	Gary Graham	Initial draft.
	2023-08-24	James Cabral.	Revisions throughout to add references and conform to OASIS best practices. Moved example schema and WSDLs in Appendix D to separate files.
	2023-08-26	James Cabral Gary Graham	Defined “RFR” in Acronyms. Corrected use “RvFR” in Section 3.12.1. Fixed typos.
1.0	2023-09-29	James Cabral Gary Graham	Minor changes to 4.6, 4.7 and D.4. Fixed typos in 3.4.3 and 3.8.2.1.
	2023-10-16	James Cabral	Added special thanks section.

Appendix F. Core Specification Normative Requirements

The following table provides a listing of all normative statements from the **[ECF-v4.1]** specification:

No.	Section/Line #	Requirement
1	2.1	In order to be compliant, an implementation of the ECF specification MUST implement the core specification and at least one service interaction profile and one document signature profile.
2	2.2	An ECF 4.1-compliant implementation may implement one or more of the MDEs defined in the specification but a complete ECF 4.1 system MUST include at least one each of the Filing Assembly, Filing Review and Court Record MDEs.
3	2.2	When multiple MDEs are implemented by a single court, vendor or application, the application MUST maintain the ECF 4.1 specified operations between each MDE so that other applications will be able to interoperate with it.
4 ?	2.2	In order to be compliant with ECF 4.1, an MDE MUST support all messages required for that MDE.
5 & 6	2.4	The court MUST have only one active, authoritative version of its policies at a given time; both the human-readable and the machine-readable statements of those policies MUST have the same release dates for the court.
7	2.4	The court's human-readable and machine-readable court policies MUST each have a version numbering method associated with it.
8	2.4.1	To be compliant with the ECF 4.1 specification, each court MUST publish a human-readable court policy
9	2.4.1	human-readable court policy MUST include each of the following: 1. The unique court identifier 2. The location of the machine-readable court policy 3. A definition of what constitutes a "lead document" in the court 4. A description of how filer identifiers are to be maintained during electronic communications regarding the case 5. A description of how the court processes (dockets) filings 6. A description of any instances in which the court will mandate an element that the ECF 4.1 schema makes optional 7. A description of any restrictions to data property values other than code list restrictions. (This restriction may be removed in later versions of the ECF specification) 8. Any other rules required for electronic filing in the court
10	2.4.2	The machine-readable court policy MUST be provided to the Filing Assembly MDE either by the Filing Review MDE through the GetCourtPolicy query or some other means.
11	2.4.5	If court-specific constraint schemas are used, instance documents MUST validate against both the ECF schemas and the court constraint schemas.

12	3.1	The operations in bold are required and MUST occur in every successful filing as long as sending and receiving MDEs are implemented in separate systems.
13	3.2	Successful queries MUST return an <ecf:ErrorCode> of “0”.
14	3.2	Failed queries MUST NOT return an <ecf:ErrorCode> of “0”
15	3.2.2	the Court Record MDE MUST have access to the court’s registry with all updated information about case participants.
16	3.2.2	There MUST be only one such registry per court,
17	3.2.2	If the court provides a Hub Service MDE, the electronic service information returned from this query MUST include the court’s Service MDE ID for all case participants who have one.
18.	3.2.4	The Filing Assembly MDE MUST submit the filing to the court by invoking the ReviewFiling operation on the Filing Review MDE.
19.	3.2.5	This operation (i.e., ServeFiling) MUST NOT be used to serve parties in a new case or to persons or organizations that have not yet been made party to the case.
20.	3.2.5	The hub Service MDE MUST then broadcast the message to each of the individual Legal Service MDE’s ServeFiling operations and respond synchronously with a single ServiceResponseMessage to the Filing Assembly MDE
21.	3.2.5	If a court chooses to support electronic service, then each Filing Assembly MDE MUST support service operations for the clients for which it provides Filing Assembly functionality.
22.	3.2.6	If the clerk reviews and accepts the filing, the Filing Review MDE MUST invoke the RecordFiling operation on the Court Record MDE.
23.	3.2.7	If the <RequireAsynchronousResponsesIndicator> in the court policy is “true”, the Court Record MDE MUST invoke the NotifyDocketingComplete operation on the Filing Review MDE as a callback message to the RecordFiling operation
24.	3.2.7	If the Court Record MDE rejected the filing, an explanation MUST be provided.
25.	3.2.7	If the Court Record MDE accepts the filing, the docketing information (e.g. date and time the document was entered into the court record, judge assigned, document identifiers and next court event scheduled) MUST be provided.
26.	3.2.8	If the clerk rejects the filings or the Filing Review MDE receives the NotifyDocketingComplete message and the <RequireAsynchronousResponsesIndicator> in the court policy is “true”, the Filing Review MDE MUST invoke the NotifyFilingReviewComplete operation on the Filing Assembly MDE
27.	3.2.8	The operation MAY return the filed documents or links to the documents, but MUST include the [FIPS 180-4] SHA 256 document hash,
28.	3.2.8	If the filing included a payment, and the filing was accepted by the clerk and court record system, a receipt for the payment MUST be included in the operation.
29.	3.3.1.1	Attachment identifiers MUST be unique within a message transmission.

30.	3.3.1.2	Case identifiers (case numbers) are assigned by the court record system and MUST be unique within a court.
31.	3.3.1.3	Court identifiers are locally assigned by the court administrator for a region (typically a state, provincial or federal court administrator) and MUST be universally unique to a court but not necessarily to a particular court house, branch or subunit of a court.
32.	3.3.1.3	Court identifiers MUST conform to following convention: <Internet domain of the court administrator>:<unique identifier within the court system>.
33.	3.3.1.4	Document identifiers are assigned by the court record system and MUST be unique within a court.
34.	3.3.1.5	Filing identifiers MUST be unique within a court and will be generated by the court in response to a ReviewFiling operation.
35.	3.3.1.6	The address of an MDE MUST be unique within a given communications infrastructure.
36.	3.3.1.7	If the <RequireAsynchronousResponsesIndicator> in the CourtPolicyResponseMessage is “true”, then both <SendingMDELocationID> and <SendingMDEProfileCode> MUST be included in all ECF 4.1 messages that include these elements.
37.	3.3.1.8	Identifiers for filers and parties to a case, both persons and organizations, MUST be unique within a case and will be generated by the court in response to a ReviewFiling operation.
38.	3.3.3.1	A CoreFilingMessage MUST express the name or names of the party or parties on whose behalf a document is filed, and the party whose document is the subject of a responsive document being submitted for filing.
39.	3.3.3.1	If a CoreFilingMessage includes documents, the message MUST include only one level of connected and supporting documents.
40.	3.4	All ROA (Record on Appeal) transactions, either the original filing or subsequent amendments, MUST contain, as the lead document, an Index of Record document that itemizes the content of the record on appeal.
41.	3.4	All ROA documents being submitted, including the Index of Record document and each document within the record, MUST have at least one court-defined document type that indicates the type of transaction to be performed on the document, and whether the document is being added to or stricken from the record.
42.	3.4	When a document within the ROA transaction is being stricken from the court record, the document MUST be identified by the unique document identifier, which was provided by the Court Record MDE when the document was initially filed (See section 3.3.1.4).
43.	3.4	A hierarchical structure of case lineage elements MUST be used to express the target case’s predecessor cases at prior courts. Each predecessor case MAY also have its own predecessor case, as necessary to express the full lineage of an appellate case.
44.	3.4	When the ROA transaction is electronically transferred from one court to another, the target case number in the destination court and the case lineage, which includes the predecessor case number in the sending court, MUST be provided.

45. 46.	3.4	If the ROA transaction is a case initiating filing in the destination court, then the <FilingCase> object MUST be present and the <CaseTrackingID> MUST be absent.
47.	3.4	If the ROA transaction is a case initiating filing in the destination court, then the <FilingCase> object MUST be present and the <CaseTrackingID> MUST be absent.
48.	3.4	When a ROA amendment transaction is sent, the Index of Record document MUST reflect the status of the record assuming that the transaction will be accepted.
49.	3.4	Individual documents within the ROA transaction MUST not be individually accepted or rejected.
50.	3.4	All documents within the ROA transaction MUST have the same acceptance or rejection disposition.

Appendix G. Notices