CDA Interoperability with SDA

Version 2024.1
2024-05-02

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

CDA Interoperability with SDA
InterSystems Version 2024.1 2024-05-02

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

List of Tables

1

CDA Documents and XSL Transforms

This page provides an introduction to the way InterSystems products handle CDA documents, beginning with an overview of the structure of a CDA document, then describing the library of XSLTs (XSL transformations) that turns CDA documents into SDA, and vice versa.

1.1 CDA Document Structure

The root node of all CDA documents is `<ClinicalDocument>`. Beneath it are three logical divisions: a header, one or more `Sections`, and one or more `Entries` within each `Section`.

- The *header* contains metadata, patient demographics, and provider information.
- A *Section* establishes a broad concept, such as Allergies or Medication. A `Section` may contain unstructured, "narrative" data in addition to structured data in the form of `Entries`.
- An *Entry* is embedded within a `Section` and represents an individual instance within the larger concept. For example, one `Entry` in the Allergies section might represent a peanut allergy, and another a pollen allergy.

When a CDA document is transformed, `Sections` and `Entries` receive transformation instructions from support files, which are discussed in XSLT Directory Structure for CDA Documents.

All portions of a CDA document conform to templates, which are defined by IHE or some other organization. These templates are identified by OIDs (object identifiers). Templates provide structured, reusable formats for clinical data in a CDA document, and they also indicate the specifications with which a CDA document must comply. Templates can inherit from one another, imposing further constraints. OIDs are made up of strings of integers separated by periods, such as `2.16.840.1.113883.3.88.11.83.102`. For more information about a specific template, enter its OID into an Internet search engine; there are many online resources that provide detailed OID specifications.

Each `Section` contains structured and unstructured data. Unstructured data contains items such as text, numbers, and even entire paragraphs; these are located in the narrative portion of a `Section`. Unstructured data accomplishes two tasks. First, it provides the human-readable section of the CDA required by CDA specifications. Second, unstructured data provides a reference point to which subsequent structured data may refer.

Structured data appears within `Sections` as `Entries`. As the name suggests, structured data has more specifications to which it must adhere. Each `Entry` has one or more templates associated with it, indicating the standards to which that `Entry` conforms. `Entries` contain a variety of data, including dates, intervals, strings, and OIDs. Additionally, they contain coded data, and particular fields may expect certain input patterns. A codeSystem attribute, for example, must be a valid OID and cannot be blank.

The following table describes the CCDA v2.1 `Sections` supported by InterSystems. The left column displays XSLs, while the right column displays their corresponding section names.

XSL	CCDA Section	Notes
AdvanceDirectives.xsl	Advance Directives	
AllergiesAndOtherAdverseReactions.xsl	Allergies and Intolerances	
AssessmentAndPlan.xsl	Assessment and Plan	
Assessments.xsl	Assessment	
ChiefComplaint.xsl	Chief Complaint	Supported for export only.
ChiefComplaintAndReasonForVisit.xsl	Chief Complaint and Reason for Visit	Supported for export only.
CarePlan.xsl		This XSL defines SDA elements for: • *Care Plan Type* • *Care Plan Provider* • *Care Plan Support Contacts* • *Care Plan SetId* • *Care Plan Version* • *Care Plan Authors* • *Care Plan Organizations* • *Care Plan Health Concern IDs* • *Care Plan Goal IDs*
DiagnosticResults.xsl	Results	
DischargeDiagnosis.xsl	Discharge Diagnosis	
EncounterDiagnoses.xsl		Used to supplement the default Encounter `section` specification.
FamilyHistory.xsl	Family History	
FunctionalStatus.xsl	Functional Status	
Goals.xsl	Goals	
HealthConcerns.xsl	Health Concerns	
HistoryOfPastIllness.xsl	History of Past Illness	
HistoryOfPresentIllness.xsl	History of Present Illness	
HospitalAdmissionDiagnosis.xsl	Admission Diagnosis	
HospitalCourse.xsl	Hospital Course	Supported for export only.
HospitalDischargeInstructions.xsl	Hospital Discharge Instructions	
HospitalDischargeMedications.xsl	Discharge Medications	
Immunizations.xsl	Immunizations	
Instructions.xsl	Instructions	

XSL	CCDA Section	Notes
Interventions.xsl	Interventions	
Medications.xsl	Medications	
MedicationsAdministered.xsl	Medications Administered	
Non-RatifiedSections.xsl		This XSL defines an SDA element for Care Considerations, including the following sub-fields: • *Entered By* • *Entered At* • *Entered On* • *External ID* • *Document Time* • *Document Number* • *Document Name* • *Document Type* • *File Type* • *Document Stream* • *Document Status* • *Clinician* • *Custom SDA Data*
Outcomes.xsl	Health Status Evaluations and Outcomes	
Payers.xsl	Payers	
PhysicalExams.xsl	Physical Exam	Supported for export only.
PlanOfTreatment.xsl	Plan of Treatment	
ProblemList.xsl	Problem	
ProceduresAndInterventions.xsl	Procedures	
ReasonForReferral.xsl	Reason for Referral	Supported for export only.
ReasonForVisit.xsl	Reason for Visit	
SocialHistory.xsl	Social History	
VitalSigns.xsl	Vital Signs	

1.2 XSLT Directory Structure for CDA Documents

InterSystems healthcare products ship with a library of XSLTs to transform CDA documents into SDA, and vice versa.

To view the available root-level XSLTs, navigate to your installation directory and follow the path *install-dir*\CSP\xslt\SDA3. This directory contains the XSLTs, many of which are named for the actions they perform.

For example, CCDA-to-SDA transforms Consolidated CDA 1.1 CCD into SDA, CCDAv21-to-SDA transforms Consolidated CDA 2.1 CCD into SDA, SDA-to-C32v25 transforms SDA into C32, and so on. These files in turn make calls to other files located in the CDA-Support-Files directory, located at *install-dir*\CSP\xslt\SDA3\CDA-Support-Files.

Each root-level XSLT (for example, CDA-to-SDA.xsl) begins with `xsl:include` declarations to pull in the appropriate files located in the CDA-Support-Files directory.

This section describes the various directories within CDA-Support-Files and the functions they serve. As the image indicates, CDA-Support-Files contains the directories:

- System — Source-controlled, non-configurable files defining widely used items such as OIDs and templates.
- Site — Configurable files that are used by various XSLTs.
- Import and Export — Files that are called when transforming CDA into SDA, and vice versa.
- Reports — Mostly transformations that turn CDA documents into HTML so that they can be displayed in a web browser. This directory is not described below.

Important: XSLTs are cached, so you must restart the production in the applicable namespace after editing a transformation for the changes to take effect.

1.2.1 The System Directory

The System directory contains static files that define a wide range of items. Items in this folder are not configurable.

The directories within System are:

- Common — Utility templates that are not widely used.
- OIDs — Variables associated with the OIDs
- Site-Defaults — A source-controlled version of the Site directory. This directory is not called at runtime. For more information, see The Site Directory.
- Templates — Variables associated with the template identifiers

1.2.2 The Site Directory

The Site directory contains files that can be uniquely configured.

Important: The files in this directory are not touched upon upgrade in order to preserve customizations. After upgrading, you must manually reconcile these files with the new versions of the files in *install-dir*\CSP\xslt\SDA3\CDA-Support-Files\System\Site-Defaults, regardless of whether they have been customized. Any files that you did not customize in the Site directory must also be refreshed with the new version in Site-Defaults by manually copying the files into the directory.

The files within Site are:

- ImportProfile and ExportProfile — Configuration settings that are used during the process of importing and exporting a CDA document into and from your instance. See Import Profile and Export Profile.
- OutputEncoding — An XSLT used on export to control the encoding of the resulting CDA document. The default is UTF-8.

- Variables — Configurable variables used during import or export. These variables represent organizations and set up "home" information.

1.2.2.1 Import Profile

The import profile controls configuration settings when importing a CDA document. Almost every variable in the import profile has a <sectionTemplateId>, an <entryTemplateId>, or both; the exception is *resultsImportConfiguration*, which instead has separate template IDs for <sectionC32TemplateId> and <sectionC37TemplateId>. The presence of a <sectionTemplateId> or an <entryTemplateId> depends on whether the variable in question is found in a section module or an entry module.

Additionally, some variables may contain other settings. The following tables list those settings as well as the variables to which they belong and their values.

Table 1–1: Import Profile Settings

Variable Name	*Setting*	*Value*
generalImportConfiguration/ blockImportCTDCodeFromText	disabled	Blocks the import of a CDA string, narrative text, or originalText into an SDA CodeTableDetail Code property when the CDA @code attribute is not available. If this setting is activated and the coded element is nullFlavor, no text is loaded into the SDA Code Property unless the target SDA element is `OrderItem` and the `orderItemDefaultCode` or `orderItemDefaultDescription` configuration parameter is turned on. If `blockImportCTDCodeFromText` is not enabled, the import behavior remains unchanged from the previous version.
generalImportConfiguration/ sdaActionCodes	enabled	Indicates whether or not SDA action codes are enabled. SDA action codes control the update and deletion of data.
generalImportConfiguration/ sdaActionCodes	overrideExternalId	Indicates whether CDA <id> elements should be used to import SDA ExternalId property values where applicable.

Variable Name	*Setting*	*Value*
generalImportConfiguration	enableOtherOrders	When enabled, and a CDA Result cannot be classified as an SDA LabOrder or RadOrder, then the Result is imported as an SDA OtherOrder. Otherwise the Result is not imported.
generalImportConfiguration/ representedOrganizationId	concatRootAndNumericExtension	When the value equals 1, if hl7:representedOrganization/hl7:id @root is an OID and @extension is numeric, then both are concatenated into one facility OID.
generalImportConfiguration	narrativeImportMode	A value of 1 imports the narrative section as text, importing both and narrative line feeds as line feeds. A value of 2 imports as text, using only as a line feed. This applies only to the import of Result Text, Hospital Discharge Instructions, and Reason for Visit.
dischargeMedicationsImportConfiguration medicationsImportConfiguration medicationsAdministeredImportConfiguration	pharmacyStatus	Indicates the status of a medication. Its value depends on the variable in which it is located, which in turn indicates the CDA section from which the medication is being imported.
planImportConfiguration	effectiveTimeCenter	When set to 0: effectiveTime/center values will be imported to FromTime. When set to 1 :effectiveTime/center values will be imported to FromTime and ToTime. Note: If effectiveTime/center is populated for a particular care plan, effectiveTime/high and effectiveTime/low should not be populated for that care plan.
resultsImportConfiguration	resultOrganizerTemplateId	Helps to select the correct hl7:organizer within a given results entry when there is more than one. One alternate value it might be given is $ihe-PCC-LabBatteryOrganizer.

Variable Name	*Setting*	*Value*
resultsImportConfiguration	orderItemDefaultCode	Code to use for SDA OrderItem Code when a CDA Result does not include information from which to derive an OrderItem Code or Description.
resultsImportConfiguration	orderItemDefaultDescription	Description to use for SDA OrderItem Code when a CDA Result does not include information from which to derive SDA OrderItem Code or Description.
encountersImportConfiguration	healthFundImportMode	A value of 1 means that this setting is enabled. This is the default. The system adds a Health Fund streamlet for each Encounter to the resulting SDA. A value of 0 means that this setting is disabled. The system will not create a Health Fund streamlet for every Encounter.
payersImportConfiguration	memberEnrollmentImportMode	A value of 1 means that this setting is enabled. For each Payor received in the CCD/CCDA, the system will create a Member Enrollment streamlet in the resulting SDA. A value of 0 means that this setting is disabled. This is the default. The system will not create a Member Enrollment streamlet in the resulting SDA.
socialHistoryImportConfiguration	patientGenderIdentityImportMode	A value of 0 means that Gender Identity Social History entries will not be imported into the GenderIdentity property of the Patient streamlet. This is the default. A value of 1 means that Gender Identity entries will be imported into the GenderIdentity property of the Patient streamlet. This setting only applies to CCDA v1.1 and CCDA v2.1.

Variable Name	Setting	Value
socialHistoryImportConfiguration	socialHistoryGenderIdentityImportMode	A value of 1 means that the system imports Gender Identity as a SocialHistory streamlet during the translation of Social History CCDA entries into SDA3. This is the default. A value of 0 means that the system does not import Gender Identity as a SocialHistory streamlet. This setting only applies to CCDA v1.1 and CCDA v2.1.

The settings in the following table control how CCDA v2.1 Note sections are imported into SDA Document streamlets. These settings apply only to CCDA v2.1.

Table 1–2: CCDA v2.1 Note Section Import Profile Settings

Variable Name	Setting	Value
notesImportConfiguration	includesections	Limit note import to sections matching the given LOINC codes. The default version of the file lists many default note sections to import. Anything that is not included is not imported.
notesImportConfiguration	excludesections	Exclude note import from sections matching the given LOINC codes.
notesImportConfiguration	includeNotes	Always import note entries matching the given LOINC codes even if the corresponding section isn't imported. The system currently does not import some Note sections, like Review of Systems, as there are no corresponding .xsl files in *install-dir*\CSP\xsl\SDA3\CDA-SupportFiles\ImportSection-Modules\CCDAv21 to import them with. If you include the LOINC code for such sections in includeNotes, any <text> blocks will be imported as a document in the SDA identified by that LOINC code. Entry data is not imported in this case. This setting is empty by default.
notesImportConfiguration	excludeNotes	Never import note entries matching the given LOINC codes even if corresponding section is imported. This setting is empty by default.

The settings in the following table control how narratives, which appear inside of CCDAv2.1 sections in <text> tags, are imported from CCDA v2.1 to SDA. These settings apply only to CCDAv2.1.

The narrativeImportConfiguration/section variable contains a list of CCDA v2.1 sections, like Admission Medication, that narratives will be imported from.

Table 1–3: CCDA v2.1 Narrative Import Profile Settings

Variable Name	*Setting*	*Value*
narrativeImportConfiguration	wrapWidth	Controls how lines are wrapped in the resulting SDA <NoteText> when narratives are imported. By default, long lines are wrapped at 80 characters. Words that end lines are broken on whitespace only. A known issue with wrapWidth prevents it from properly wrapping narrative text that has been imported into SDA.
narrativeImportConfiguration	exclude	Optional. Any narrative that contains a phrase included in the exclude setting is skipped for import. Any narratives that exactly match any phrases inside of the <exclude mode="full"> option will be skipped for import. Any narrative with the nullFlavor attribute is always skipped. The importProfile.xsl file contains several default phrases in <exclude>. You can either exclude phrases via this exclude setting, which filters all the sections that you are importing narratives from, or you can use the section-specific exclude setting described below to exclude from a specific section.
narrativeImportConfiguration/section	title	Recommended. This setting affects how the SDA is displayed in the Clinical Viewerand will translate into the SDA as DocumentName in the SDA Document streamlet, which is used for Clinical Viewer grouping. More specifically, this value overrides the hl7:section/hl7:title value that is mapped to DocumentName.
narrativeImportConfiguration/section	code	Optional. Affects how the SDA is displayed in the Clinical Viewer. This setting becomes the Code in the DocumentType. The displayName option of code becomes the Description of the documentType property of the SDA Document streamlet.

Variable Name	Setting	Value
narrativeImportConfiguration/section	templateId	Required. Every category for clinical data has a defined unique templateID, or OID. If present, the narrative for the section specified by this templateId is imported.

1.2.2.2 Export Profile

Most elements in the export profile have an <emptySection> element, a <narrativeLinkPrefixes> element, or both. <emptySection> elements control whether or not to export a section that contains no information. <narrativeLinkPrefixes> allows you to determine the prefix of the IDs that will identify narrative sections (which are located inside of <text> tags), and entry sections in the resulting CCDA. These IDs can be used to provide links between the narrative and entry sections in the resulting CCDA. Both of the following have <narrativeLinkPrefixes>, as both result in sections that require links between narrative and entry sections:

- Top-level ExportProfile.xml sections like <allergies> and <assessmentPlan>
- The <notes> section used for Note section export in ExportProfile.xml

Additionally, some sections may contain other settings. These settings can be edited and augmented depending on the data that is to be exported. The following tables list those unique settings as well as the sections in which they are found and their values.

Table 1–4: Export Profile Settings

Section	Setting	Value
admissionDiagnoses/ diagnosisType	codes	Codes that can be used as additional diagnosis types for admission.
advanceDirectives/ advanceDirectiveType	codes	Codes that correspond to various advance directive types, such as resuscitation or intubation.
assessment/ diagnosisType	codes	Codes that can be used as additional diagnosis types for assessment.
dischargeDiagnoses/ diagnosisType	codes	Codes that can be used as additional diagnosis types for discharge.
encounterDiagnoses/ exportToC32	disabled	If enabled (value = 1), allows the export of encounter diagnoses to CDA C32.
medications/ currentMedication	includeHistoricalMedications	Controls whether or not to include historical medications in the current medications list.
medications/ currentMedication	windowInDays	Limits how old in days a medication can be and still be included in the current medications list.

Section	*Setting*	*Value*
medications/ currentMedication	hideNarrativeColumn	Hides the narrative column.
planOfCare	effectiveTimeCenter	When set to 0: FromTime will be exported to effectiveTime/low and ToTime will be exported to effectiveTime/high. When set to 1: IF FromTime and ToTime have the same non-null value, that value is exported to effectiveTime/center. ELSE: if FromTime has a value, it is exported to effectiveTime/low. If ToTime has a value it is exported to effectiveTime/high. EXCEPTION: if ProcedureTime has a value, it is exported to effectiveTime/Center in place of FromTime.
problems/ currentCondition	codes	Codes that correspond to various problem types, such as `ACTIVE` or `CHRONIC`.
problems/ currentCondition	windowInDays	Limits how old in days a medication can be and still be included in the current problems list.
socialHistory/ emptySmokingStatus	exportData	Indicates whether or not to export smokingStatus when it contains no data.
exportConfiguration/ Immunizations	checkAdministrations	Included for backwards compatibility, so that users can successfully process SDA Vaccinations and create CDA/C-CDA data with Immunizations if they have both: • Erroneous SDA with Immunization administration dates translated incorrectly in SDA, resulting in SDA without Administrations (if the SDA was generated prior to receiving the CDA/C-CDA Immunization Processing Correction) • Newer SDA without the issue, where Immunization administration dates were correctly translated to Vaccination.Administrations.Administration.FromTime When set to 0: This is the default. Immunizations entries are only created when Vaccination SDA correctly includes Administrations. When set to 1: Immunization entries are created in CDA/C-CDA for SDA Vaccinations created both before and after receiving the CDA/C-CDA Immunization Processing Correction. Note that this setting may be unavailable in your version. If you need this functionality, contact the WRC and request an ad hoc.

The settings in the following table control how SDA Document streamlets, which represent CCDA Note sections like Progress Note and Procedure Note, are exported to CCDA v2.1 Note sections. These settings apply only to CCDA v2.1.

The notes/category section contains settings that describe an SDA Document category for export to CCDA v2.1 Note sections. Including a category section for a specific SDA Document type exports that specific SDA Document from SDA to CCDA v2.1.

Table 1–5: CCDA v2.1 Note Section Export Profile Settings

Section	*Setting*	*Value*
notes	singleSection	A value of 0 means that each category specified is exported into its own separate section in the resulting CCDA v2.1 document. This is the default. Setting this value to 1 will instead export all notes into a single default section as one single note. The default section is specified by the *default* setting described later in this table.
notes	uncategorized	A value of 0 means that SDA documents that don't match a category specified in the notes/category section are not exported to CCDA v2.1. Setting this value to 1 will instead export these documents to the default section, specified later in this table by the *default* setting.
notes	default	Must be defined. Specifies the category that will be used as the default notes section when singlesection and uncategorized are set to 1.
notes	code	The LOINC code for the document type. Used at the time of export to match with the correct note category.
notes/category	name	The LOINC display name for the document type.
notes/category	title	The section title to use.

Section	Setting	Value
notes/category	typecodes	A pipe-delimited list of LOINC codes that fall under this SDA Document category. Each typecode is a LOINC code representing a subcategory of the current Document category. These LOINC codes are subcategories of the CCDA Note section that is represented by the Document category. The system will put the types of notes that are represented by these LOINC codes inside of the overarching Note section in the resulting CCDAv2.1. For example, you might enter several typecodes, like `|11526-1|34122-2|34819-3|`, that are part of a Pathology Study. All of the notes represented by these typecodes would end up inside of the Pathology Study Note in the resulting CCDA v2.1. Any LOINC code that is not included is not exported.

The settings in the following table control how narratives that appear inside of CCDAv2.1 sections in <text> tags are exported from SDA to CCDA v2.1. These settings apply only to CCDAv2.1. The <narrative> variable contains a list of sections that will have narratives exported. Each <section> contains the same settings as a narrativeImportConfiguration/section that is used for narrative import above.

Note that the default ExportProfile.xml file divides <narrative> into three types of sections:

- Standard sections — Sections for which the system has a .xsl file for exporting from SDA to CCDAv2.1 in *install-dir*\CSP\xslt\SDA3\CDA-Support-Files\Export\Section-Modules\CCDAv21. The system exports both narratives and entries for these sections.
- Special sections — Sections for which the system previously exported both entries and narratives. In this case, the system defaults back to the legacy export logic.
- Standalone sections — Sections for which InterSystems IRIS for Health does not have a .xsl file for exporting the section in *install-dir*\CSP\xslt\SDA3\CDA-Support-Files\Export\Section-Modules\CCDAv21, meaning that neither entries nor narratives are exported. If a section is included in the <narratives> variable as a standalone section, the system will export the narratives in the section without exporting the entries. Many sections that are supported for narrative import are not supported for narrative export.

The import and export profiles control which sections are eligible for narrative processing. Any section is eligible for import. However, only a subset of standard sections are enabled for export - for example the Allergy section is not included for export, but the FunctionalStatus section is. In most cases, the sections that are included by default tend to include relevant data in narratives that is not referenced by entries.

Table 1–6: CCDA v2.1 Narrative Export Profile Settings

Section	Setting	Value
narrative/section	title	Used as a separator heading after entry-based narrative if present in a standard or standalone section. Not used in the resulting CCDA in a special section.
narrative/section	code	Informative only and is not used in the resulting CCDA for all three types of sections.
narrative/section	templateId	Required. Every category of clincial data has a unique templateId, or OID. The narrative for the specified section is exported.

1.2.2.3 Adding Additional Standalone Sections for Narrative Export

To export additional standalone sections that are not already included in the list of standalone sections to be exported in the narrative/section section of ExportProfile.xml, do the following:

1. Add the appropriate <section> inside of the <narrative> section in ExportProfile.xml, for example:

```
<section>
<title>Admission Medication</title>
<code code="42346-7" displayName="Medications on Admission"/>
<templateId root="2.16.840.1.113883.10.20.22.2.44" extension="2015-08-01/>
</section>
```

2. If you have not already done so, create a custom version of the appropriate root-level .xsl file used for SDA to CCDA v2.1 export in *install-dir*\CSP\xslt\SDA3\CDA-Support-Files\Export\Entry-Modules\CCDAv21 and create a custom version.

   In most cases, you will be editing the SDA-to-CCDAv21-CCD.xsl file, as narrative export is only supported for CCDA v2.1. Occasionally, depending on what transform you use to export CCDA, you may need to edit another file, like SDA-to-CCDAv21-ClinicalSummary.xsl or SDA-to-CCDAv21-CON.xsl.

3. In your custom version of the root-level .xsl file, add a line like the following in the `<xsl:apply-templates mode="narrative-export-sections" select=".">` section:

   ```
   <item root="{$ccda-PhysicalExamSection}"/>
   ```

   The `<xsl:apply-templates mode="narrative-export-sections" select=".">` section of the .xsl file specifies which standalone narratives to include during the export. You can determine the name of the variable (for example *$ccda-PhysicalExamSection*) that you need to add by examining the TemplateIdentifers-CCDA.xsl file in the *install-dir*\CSP\xslt\SDA3\CDA-Support-Files\System\Templates directory and finding the variable name that corresponds to the section that you wish to export.

   Adding this line will call the narrative-export-sections template on each item in the `templates` parameter, which is necessary for narrative export.

1.2.2.4 Adding Additional Standard Sections for Narrative Export

Certain standard sections support narrative export by default. In order to enable narrative export for other standard sections, you must customize the .xml files for that section in the Section-Modules and Entry-Modules subdirectories. The following provides an example of how you would enable narrative export for the FunctionalStatus standard section, though FunctionalStatus is enabled for narrative export by default:

1. Add the appropriate <section> inside of <narrative> in ExportProfile.xml.
2. Navigate to the *install-dir*\CSP\xslt\SDA3\CDA-Support-Files\Export\Section-Modules directory and open the FunctionalStatus.xml file for editing.
3. Define a variable to hold the node-set of SDA Documents for the section:

   ```
   <xsl:variable name="docs" select="Documents/Document[(Category/Code/text() = 'SectionNarrative')
   and (DocumentType/Code/text() = $exportConfiguration/narrative/section[templateId/@root =
   $ccda-FunctionalStatusSection]/code/@code)]"/>
   ```

   This uses the export configuration to check if the section is enabled for structured narrative export.
4. Define another variable to indicate if there are eligible docs:

   ```
   <xsl:variable name="hasDocs" select="count($docs)"/>
   ```

5. Update any conditionals that check if the section has data to use your new *hasDocs* variable, for example:

   ```
   <xsl:if test="($hasDocs > 0) or ($hasData > 0) or ($exportSectionWhenNoData='1') or
   ($sectionRequired='1')">

   <xsl:if test="($hasDocs = 0) and ($hasData = 0)"><xsl:attribute
   name="nullFlavor">NI</xsl:attribute></xsl:if>
   ```

6. Update the call to the Entry-Module template to pass the *docs* variable:

   ```
   <xsl:apply-templates select="." mode="eFS-functionalStatus-Narrative">
      <xsl:with-param name="docs" select="$docs"/>
   </xsl:apply-templates>
   ```

7. Add a `xsl:when` clause to export the section when there are no entries but there are SDA Documents available:

   ```
   <xsl:when test="$hasDocs > 0">
      <text>
         <xsl:apply-templates mode="narrative-export-documents" select=".">
            <xsl:with-param name="docs" select="$docs"/>
         </xsl:apply-templates>
   </text></xsl:when>
   ```

8. Navigate to the *install-dir*\CSP\xslt\SDA3\CDA-Support-Files\Export\Entry-Modules\CCDAv21 directory and open the FunctionalStatus.xml file for editing.
9. Add the *docs* parameter to the narrative template:

   ```
   <xsl:template match="*" mode="eFS-functionalStatus-Narrative">
      <xsl:param name="docs"/>
   ```

10. Call the section exporter after the standard entry table:

    ```
    <text>
       <table border="1" width="100%">
          ...
       </table>

       <!-- structured narrative documents -->
       <xsl:apply-templates mode="narrative-export-documents" select=".">
          <xsl:with-param name="docs" select="$docs"/>
          <xsl:with-param name="header"
    select="$exportConfiguration/narrative/section[templateId/@root=$ccda-FunctionalStatusSection]/title/text()"/>

       </xsl:apply-templates>
    </text>
    ```

1.2.3 The Import and Export Directories

The Import and Export directories contain files that are called when XSLTs transform data.

- The Import directory contains those files that are called when transforming CDA into SDA.
- The Export directory contains those files that are called when transforming SDA into CDA.

Both the Import and the Export directories contain subdirectories labeled Common, Entry-Modules, and Section-Modules.

- The Common directory contains XSLTs that set up commonly used global variables for the various transformations and provide templates that contain commonly used logic, such as `address-Home`.
- The Section-Modules directory contains XSLTs that transform data to or from a given CDA section; the name of each XSLT corresponds closely to the name of the CDA section it transforms.
- The Entry-Modules directory contains XSLTs that map the transformation into the correct `Entry` of a document and then parse the coded data. As a result, XSLTs within this directory tend to be significantly longer than their counterparts in the Section-Modules directory.

Each root-level XSLT (for example, CDA-to-SDA.xsl) begins with `xsl:include` declarations to pull in the appropriate Section-Modules, Entry-Modules, and Common files, as well as other necessary files.

2

Customizing CDA XSL Transformations

You may occasionally need to augment the standard library of CDA XSL transformations. For example, you may need to change part of the logic when transforming a particular CDA section, or you may wish to add support for custom SDA extensions, custom pairs, or custom objects. For more information on custom SDA extensions, pairs, and objects, see the Customizing the SDA section of *SDA: InterSystems Clinical Data Format*.

Instead of editing the standard library of XSLs directly, you can create custom XSLs. This is preferable to direct editing for two reasons. First, overriding a standard root XSL in this manner negates the need to change existing application calls to that transformation. Second, it provides insulation against the effects of upgrades, which replace the standard XSL library and thereby erase your customizations. XSLs in a custom directory are not replaced upon upgrade.

2.1 Preparing to Customize Transformations: Docker Containers

Important: If you plan to customize transformations and InterSystems IRIS for Health is running in a Docker container, the container must point to a durable directory. For more information on durable directories, see *Running InterSystems Products in Containers*.

If your instance is running within a container, you must take the following preparation steps before you start to customize the transformations:

1. In the Management Portal, create a `/csp/xslt` web application:
   a. Log in to the Management Portal as a user with the **`%HS_Administrator`** role.
   b. Navigate to **System Administration** > **Security** > **Applications** > **Web Applications**.
   c. Click **Create New Web Application**.
   d. In the form, complete the fields as follows:
      - **Name** — **`/csp/xslt`**
      - **Namespace** — HSLIB
      - **Enabled** — **Enable Application** and **CSP/ZEN** should be selected
      - For **CSP File Settings:**
        – **Serve Files** — accept the default
        – **Server files timeout** — accept the default

- **Physical Path** — a directory within your durable directory, such as `/container/durable/csp/xslt/`. Make sure to include the final trailing slash.
- **Web Settings** — accept the defaults

e. Click **Save**.

2. Open a Terminal session and log in as a user with the **`%HS_Administrator`** role.
3. To copy the XSLT files for customization to the physical path specified in the step above: within the **`HSLIB`** namespace, run

ObjectScript

```
do ##class(HS.HC.Util.Installer.Upgrade.XSLTDirectoryCopy).Update()
```

To see verbose output, you can supply `1` as the single parameter for the `Update()` method.

2.2 Creating Custom Transformations

To create a custom CDA transformation:

1. Create a Custom subdirectory in the directory where the transformation is located. For example, if you are customizing *install-dir*/CSP/xslt/SDA3/CDA-to-SDA.xsl, the custom subdirectory should be *install-dir*/CSP/xslt/SDA3/Custom. Files in a custom folder are not overwritten upon upgrade.
2. In the IDE of your choice, open the transformation you intend to customize so you can copy portions of it.
3. Create an empty custom XSL with the same name as the XSL you intend to customize. For example, if you want to customize CDA-to-SDA.xsl, create an XSL in *install-dir*/CSP/xslt/SDA3/Custom called CDA-to-SDA.xsl.
4. Add the following initial statements to your custom XSL. The first two (the XML declaration and the XSL stylesheet) can be copied from the original CDA XSL file.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:isc="http://extension-functions.intersystems.com" xmlns:hl7="urn:hl7-org:v3"
 xmlns:sdtc="urn:hl7-org:sdtc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:exsl="http://exslt.org/common" exclude-result-prefixes="isc hl7 sdtc xsi exsl">
```

5. Add a closing tag for the stylesheet. Your customizations will all be placed between the opening and closing `xsl:stylesheet` tags.

```
</xsl:stylesheet>
```

6. Add an import statement for the transformation you intend to customize:

```
<xsl:import href="../<filename>"/>
```

where `<filename>` is the name of the transformation you intend to customize, for example, CDA-to-SDA.xsl. This imports the file you intend to customize so that you can override one or more of its templates.

7. Copy the template you wish to override from the original transformation and paste it into the new file.
8. Customize the template logic as necessary. This overrides the template in the original XSL.
9. As a best practice, add a comment to indicate where the template comes from. For example, if the template originated in *install-dir*/CSP/xslt/SDA3/CDA-Support-Files/Import/Entry-Modules/Medications.xsl, make a note of it.

10. Save your work.
11. If your installation is deployed in a mirror, repeat the process on all mirror members.
12. Restart your productions to activate your custom XSLs.

2.3 Extending Transformations

SDA can be extended in several ways, as described in Customizing the SDA. These extensions can be incorporated into custom CDA transformations:

- Custom containers or objects can capture CDA sections that are unsupported
- Extension classes can add data that belongs in a supported section but is not captured by the SDA

Blank templates are included in CDA import and export XSLs to facilitate:

- Importing a CDA section that is unsupported into a custom container or <CustomObject>
- Importing CDA data into SDA extension classes or <CustomPairs> in existing SDA sections
- Exporting a custom container or <CustomObject> into a CDA section that is unsupported

Blank templates always contain the word `Custom` for easy identification.

The following gives details on how each of the above customizations are implemented:

- To import a CDA section that is unsupported into a custom container or <CustomObject>, find the custom template in the XSL you wish to edit. For example, in CDA-to-SDA.xsl, the template is called `<xsl:template match="*" mode="ImportCustom-Container">`. Copy that template into your custom CDA-to-SDA.xsl file (or the custom version of whichever file you are customizing) and edit the blank custom template to include whatever logic you require.
- To import CDA data into SDA extension classes or <CustomPairs> in existing SDA sections, find the custom template in the import XSL of the desired section. For example, to import the `BodySite` element into `Procedure`, find the custom template within Entry-Modules/Procedure.xsl. In this example, the template is called `<xsl:template match="*" mode="ImportCustom-Procedure">`. Copy the template into your custom CDA-to-SDA.xsl file (or the custom version of whichever file you are customizing) and edit the blank custom template to include whatever logic you require.
- To export a custom container or <CustomObject> into a CDA section that is unsupported, find the custom template in the XSL you wish to edit. For example, in SDA-to-C32v25.xsl, the template is called `xsl:template match="*" mode="ExportCustom-ClinicalDocument">`. Copy that template into your custom SDA-to-C32v25.xsl file (or the custom version of whichever file you are customizing) and edit the blank custom template to include whatever logic you require.

2.4 Debugging Custom Transformations

Debugging custom transformations consists of a set of tools and techniques that can be applied to a variety of situations.

For debugging, you can:

- Use code contained within <DEBUGGING> tags to capture a value during execution of the transformation

- Use interoperability tools to view the trace of a document within a production (HealthShare Unified Care Record only)

2.4.1 Debugging Code

<DEBUGGING> tags are a way to perform a step-by-step walkthrough of the transformation. Place the tags in the transformation at the location where you would like to capture a value. The code within the tags can capture the value of a variable, attribute, or XPath.

The following are examples of debugging code for different situations.

Capturing the value of a variable

This example captures the value of a variable, *$variable*, within an XSL called FileName.xsl:

```
<DEBUGGING-VARIABLE-file-FileName.xsl-TemplateName>
$variable = <xsl:value-of select="$variable"/>
</DEBUGGING-VARIABLE-file-FileName.xsl-TemplateName>
```

Capturing the value of an attribute

This example captures an XSL path value of the `root` attribute located in `/ClinicalDocument/id`:

```
<DEBUGGING-PATHVALUE>
CDA source OID = <xsl:value-of select="/hl7:ClinicalDocument/hl7:id/@root"/>
</DEBUGGING-PATHVALUE>
```

Capturing an XPath with Node Positions

Capturing an XPath takes advantage of templates. Calling the template `currentXPathWithPos` returns the current XPath with node positions, with `ClinicalDocument/component[1]` indicating, for example, the first `component` element inside `ClinicalDocument`. The following shows how to capture the XPath of the current position with node positions.

```
<DEBUGGING-PATHXMLWithPos>
<xsl:apply-templates select="." mode="currentXPathWithPos"/>
</DEBUGGING-PATHXMLWithPos>
```

The result of this call may be:

XML

```
<DEBUGGING-PATHXMLWithPos>/ClinicalDocument[1]/component[1]/structuredBody[1]/component[8]
/section[1]/entry[1]/procedure[1]</DEBUGGING-PATHXMLWithPos>
```

Capturing an XPath without Node Positions

Calling the `currentXPath` template returns the XPath without the node positions. The following example shows how to capture the XPath of the current position without node positions.

```
<DEBUGGING-PATHXMLWithoutPos>
<xsl:apply-templates select="." mode="currentXPath"/>
</DEBUGGING-PATHXMLWithoutPos>
```

The result of this call may be:

XML

```
<DEBUGGING-PATHXMLWithoutPos>/ClinicalDocument/component/structuredBody/component
/section/entry/procedure</DEBUGGING-PATHXMLWithoutPos>
```

The templates `currentXPathWithPos` and `currentXPath` are found in *install-dir*/csp/xslt/SDA3/CDA-Support-Files/System/Common/Functions.xsl.

Capturing an XPath tree

Calling the `copy` template returns the current XPath tree. The following example captures the current XPath tree by inserting the current XPath tree in a DEBUGGING tag:

```
<DEBUGGING-PATHXMLTree>
<xsl:apply-templates select="." mode="copy"/>
</DEBUGGING-PATHXMLTree>
```

The template `copy` is found in *install-dir*/csp/xslt/SDA3/CDA-Support-Files/System/Common/Functions.xsl

These <DEBUGGING> tags and the values they capture are displayed in the transformations, but they are not displayed in places such as the Clinical Viewer and the View Summary page.

<DEBUGGING> tags may short-circuit CDA processing. Accordingly, they should be used during the debugging process only.

2.4.2 Interoperability Tools

InterSystems products provide interoperability tools that allow you to inspect the transformation process. You can inspect the process at the following points using message traces:

- HealthShare Unified Care Record:
  - During document retrieval, in the consumer Edge Gateway production (when a CDA in the repository is transformed into SDA)
  - When an incoming SDA is received by the Clinical Viewer
- InterSystems IRIS for Health and Health Connect: examine the message trace for the relevant business operation. For example, for inbound SDA, look at the message trace for the business operation that handles the transformation for inbound SDA processing.

Additionally, you can view the transformation of an outgoing CDA using the View Summary page.

2.4.2.1 Message Traces

To view a message trace:

1. Log in to the Management Portal as a user with the **%EnsRole.Administrator** role.
2. Choose the appropriate namespace:
   - In HealthShare Unified Care Record:
     - To view the trace for document retrieval, choose the namespace for the consumer Edge Gateway.
     - To view the trace for an incoming SDA, choose the namespace for the Access Gateway.
   - In InterSystems IRIS for Health or Health Connect: choose the Foundation namespace whose business hosts handle the transformation.
3. Navigate to **Interoperability** > **Configure** > **Production**.
4. Open your production.
5. Choose the appropriate component. For example, **HS.IHE.XDSb.Consumer.Operations** might be used for document retrieval.
6. Click the **Messages** tab in the panel to the right.
7. Click the **Go To Message Viewer** link.

8. Choose the message representing the transformation. Following the example above, this message would have **`HS.IHE.XDSb.Consumer.Operations`** as its source.
9. Click the **Trace** tab in the panel to the right.
10. Click the **View Full trace** link.
11. Click the **Contents** tab in the panel to the right.

The **Contents** tab contains the message trace.

2.4.2.2 View Summary

If you are running HealthShare Unified Care Record, you can also compare CDA documents with their SDA counterparts within the Clinical Viewer.

To use the Clinical Viewer:

1. Log in to the Management Portal as a user with the **%HS.Administrator** role.
2. Navigate to **HealthShare** > <*Access_Gateway_Namespace*> **Patient Search**.
3. Enter search information for the patient whose CDA document you wish to view.
4. Select the patient.

From the Clinical Viewer, click the **View Summary** link at the top of the page to arrive at the View Summary page. Here you may view the patient summary as an SDA or as a CDA document, such as a Continuity of Care Document (CCD, HITSP C32), among other options. For more information on using the View Summary page, see Viewing and Modifying Patient Reports.

3

Preprocessing C-CDA 2.1 Documents

InterSystems healthcare products support import transformations from C-CDA 2.1 to SDA via XSLT 1.0. C-CDA 2.1 to SDA transformations via XSLT 2.0 are not supported.

Among the enhancements to import functionality added in connection with C-CDA 2.1 support is the ability to preprocess your C-CDA input files prior to the transformation done for import. Preprocessing support can greatly simplify and reduce total processing time for transformations.

Some possible use cases are:

- Missing or malformed elements
- Datestamp translation

A detailed explanation of preprocessing C-CDA to remove punctuation and to add required child nodes to certain elements, as well as sample code for a preprocessor transform, a modified top-level transform, and an input file, can be found in the article Preprocessing Support for C-CDA 2.1 Import Transformations on the InterSystems Developer Community.

In general, the steps are as follows:

1. Working with the administrator of your organization's domain names, obtain a subdomain name to be used for code extensions.
2. Choose a prefix and a name for a custom mode for preprocessing.
3. Create your XSL preprocessor.
   - The `stylesheet` element should be of the form

     ```
     <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
     xmlns:hl7="urn:hl7-org:v3" xmlns:custom_mode_prefix="<subdomain>" exclude-result-prefixes="hl7
      <custom_mode_prefix>"></xsl:stylesheet>
     ```

     Note the inclusion of the custom mode prefix.

   - Include templates that replicate the top-level element and copy all attributes, elements, and text nodes. Both templates must use your custom mode.
   - Create a template for each distinct modification you want to make to the C-CDA. These templates must also use your custom mode.
4. Copy the XSL preprocessor to `../CDA-Support-Files/Import/`.
5. Create a custom top-level transform by cloning and modifying CCDAv21-to-SDA.xsl or CCDAv21-nonXML-to-SDA.xsl as follows:
   - The `stylesheet` element should include your custom namespace.

- Add an include statement that gives the pathname for the XSL preprocessor.
- Replace the line

  ```
  <xsl:variable name="input" select="/hl7:ClinicalDocument"/>
  ```

  with the block

  ```
  <xsl:variable name="inputRTF">
     <xsl:apply-templates select="/hl7:ClinicalDocument" mode="<custom_mode_prefix>:<mode>"/>
  </xsl:variable>
  <xsl:variable name="input" select="exsl:node-set($inputRTF)/hl7:ClinicalDocument"></xsl:variable>
  ```

6. In your production, find the business service that takes file input for C-CDA > SDA transformations.
7. Set the value of the service's **`InputXSL`** additional setting to the filename of your custom top-level transform.
8. Import your C-CDA file as you normally do.

4

CDA and SDA Annotations

When a CDA document is converted to SDA, or vice versa, the relationship between fields of the source document and those of the target document can often be difficult to determine. To help untangle such relationships, the documentation mapping utility allows you to identify the source field or fields for a given target field.

4.1 Searching for Annotations

In the Management Portal, navigate to:

- **HealthShare** > <*namespace*> > **Schema Documentation** > **SDA/CDA Annotations**, where <*namespace*> is an Edge Gateway namespace for HealthShare Unified Care Record
- **Health** > <*namespace*> > **Schema Documentation** > **SDA/CDA Annotations**, where <*namespace*> is your Foundation namespace for InterSystems IRIS for Health or HealthConnect

To identify source fields, you must first generate a list of target fields from which to choose:

1. Choose one or more target document types. For example, if you wish to identify the source of a field in an SDA document, select the `SDA` check box. You may select multiple document types, or select `All Types`. The types you can choose from are:
   - *SDA* — The class and property path of a particular SDA element, for example, `HS.SDA3.Medication.Comments`.
   - *XPathSDA* — The XPath of a particular SDA element. This type can be more useful than *SDA* when looking at the XML for a container, or when writing XSLT for the XML. For example, the XPath `/Container/Medications/Medication/Comments` corresponds to the class and property path `HS.SDA3.Medication.Comments`.
   - *CCDA1* — Consolidated CDA Release 1.1.
   - *CCDA2* — Consolidated CDA Release 2.1.
   - *C32* — HITSP/C32.
   - *XDLAB* — XD-LAB, the CDA document for the IHE Laboratory Technical Framework.
2. Type a complete or partial field name or path in the text box; you may enter as much or as little information as you have. You may also paste an XML fragment; the fragment does not need to begin and end with the same tag, but the field name for which you are searching must be enclosed by angle brackets, such as `<target-field>`. This allows you to find a complex field name with simple search criteria.

3. When you have finished entering text, generate a list of possible target fields by clicking **Show Options**, or by pressing the Enter key while your cursor is in the text box.
4. Choose a target field from the drop-down menu. The menu is limited to 250 results; enter more text or choose fewer document types in order to narrow the results.

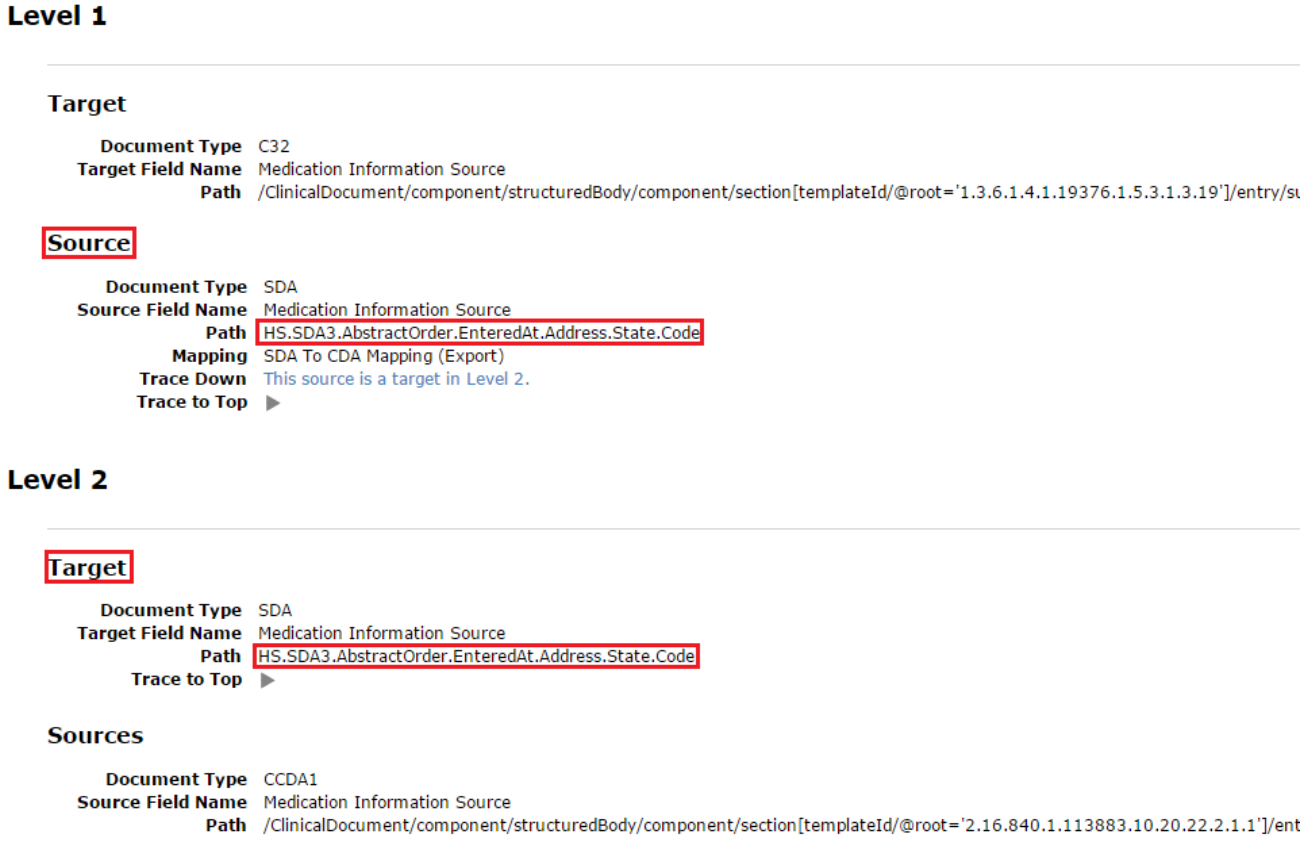
The document transformation mapping information appears below the search area; you can refocus the page by clicking the **Move Results to Top** link. The target field you selected is displayed under the **Target** heading, along with:

- The **Document Type** to which the target corresponds.
- The **Target Field Name**, if it exists.
- The **Path** of the target.

The **Source** heading also contains the document type, field name (if it exists), and path of the source field, as well as notes about the mapping and the type of mapping that is performed to transform the source into the target; for example, `HL7 to SDA3 Mapping` to turn an HL7 source into an SDA target.

4.2 Levels in Annotations

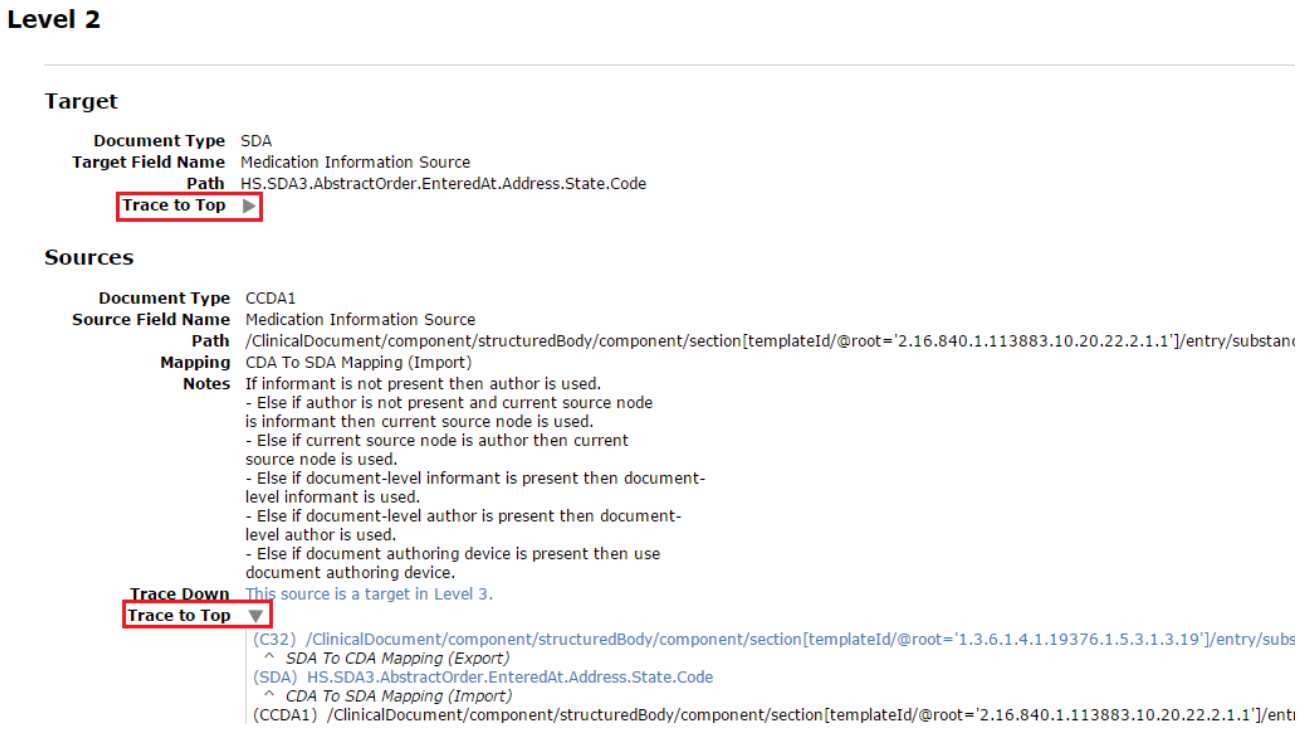
If a source field is itself the target of other fields, the mapping utility displays those fields in a new **Level**. For example, in the image below, `HS.SDA3.AbstractOrder.EnteredAt.Address.State.Code` is the source of the indicated C32 field in level 1; in turn, it is also the target of a field from a C-CDA1 document in level 2.



To navigate between levels, click the link **This source is a target in Level** `[n]` or the link **This target is a source in Level** `[n]`, where `n` is a positive integer. You can also click on a path value to highlight in color identical path values throughout all levels; click again to remove the highlighting.

Each **Target** and **Source** heading also contains a **Trace to Top** option. Click the arrow to expand it. This option displays the mapping hierarchy, where:

- The top-level target field (that is, the one you searched for) is at the top,
- The field where you are located is at the bottom,
- And any intermediate fields (if they exist) occur in between.



These hierarchical levels are connected by mappings, the direction of which are indicated by carets, `^`.

Each field in the hierarchy contains a link that focuses the page on the location of that field as a target. For example, in the image above, clicking on `(SDA) HS.SDA3.AbstractOrder.EnteredAt.Address.State.Code` would refocus the page on the **Target** section immediately above it, where that field acts as a target.