USDM Handbook 3: Leveraging USDM to Generate Define-XML (USDM-HB3 v1.0)

Version 1.0 (Final)

Prepared by the USDM Team

Notes to Readers

This is Version 1.0 of the USDM-HB3 v1.0.

Revision History

Date	Version
2026-11-16	1.0 Final

See Appendix E for Representations and Warranties, Limitations of Liability, and Disclaimers.

CONTENTS

1 INTRODUCTION 3

2 RELEVANT WORKFLOWS AND PROCESSES 5
2.1 Business Level 5
2.2 User Level 6
2.3 Technical Level 7
2.4 Maturity Matrix 9

3 SOLUTION REQUIREMENTS 10
3.1 Relevant Stakeholders 10
3.2 Upstream Requirements for the Study Design Solution 11
3.3 Downstream Requirements for the Creation of Define-XML 11
3.3.1 Functional Requirements 11
3.3.2 Non-functional Requirements 12
3.3.3 System Context and Architecture 12

4 TECHNICAL IMPLEMENTATION 15
4.1 From USDM to Define-XML: The Metadata Chain 15
4.2 Element-level Mapping 22
4.3 Supplying Study-Specific Metadata Not in the Standard Sources 23
4.4 Validation 25

5 CHALLENGES AND CONSIDERATIONS 27

6 EXAMPLES AND USE CASES 29

APPENDICES 32
Appendix A: USDM Example JSON 32
Appendix A1: Instantiated BC: Informed Consent 32
Appendix A2: Instantiated BC: Platelet Count 35
Appendix A3: Instantiated BC: Weight 41
Appendix B: Implementation Handbook Team 46
Appendix C: Glossary and Abbreviations 47
Appendix D: References 49
Appendix E: Representations and Warranties, Limitations of Liability, and Disclaimers 50

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

1 Introduction

Purpose of this Handbook

A Define-XML document is the metadata backbone of a regulatory submission: It describes the datasets, variables, controlled terminology, value-level metadata, and documentation that a reviewer needs to understand and navigate the submitted data. Historically, this metadata has been manually assembled, late in the study lifecycle, from sources that were never designed to feed it. This implementation handbook shows how a Define-XML document can instead be generated as a study specification from structured study metadata that already exists in the Unified Study Definitions Model (USDM; https://www.cdisc.org/ddf), reusing the CDISC Biomedical Concepts (BCs; https://www.cdisc.org/cdisc-biomedical-concepts) and Study Data Tabulation Model (SDTM; https://www.cdisc.org/standards/foundational/sdtm) dataset specializations (DSS) referenced in the USDM metadata together with SDTM domain metadata and controlled terminology drawn from the CDISC Library.

The emphasis of this handbook is on metadata specification, not data transformation. It describes how to assemble and generate the metadata that constitutes a Define-XML document from digital sources. It does not describe how to map or convert subject-level source data into SDTM datasets. Define-XML generation is a downstream use case in the digital data flow: It consumes the structured study definition produced upstream and turns part of it into a submission-ready metadata artifact.

For readers new to Define-XML (https://www.cdisc.org/standards/data-exchange/define-xml), CDISC offers a variety of training options, including 2 virtual classes:

- Define-XML Module 1: Introduction to Define-XML
- Define-XML Module 2: Submission Datasets and Variables

> ***Key Insight:*** *In a USDM-driven environment, the Define-XML document is not a deliverable assembled from scratch at the time of submission; it is a rendition of metadata that already exists in the study design used to generate the Define-XML document as a specification. The BCs bound to study activities identify which SDTM DSS, and therefore which value-level metadata and controlled terminology, belong in the Define-XML document. This turns specialization selection from a manual exercise into a consequence of the study design.*

The USDM Handbook Series

This handbook is the third USDM Implementation Handbook. The handbooks address successive uses of the same structured study definition.

- **USDM Handbook 1: Leveraging USDM Metadata to Automatically Construct the Foundational CDISC Trial Design Model Domains (USDM-HB1)** describes construction of the SDTM Trial Design Model domains from USDM structural metadata (arms, epochs, elements, study cells; available at https://www.cdisc.org/USDM-HB1).
- **USDM Handbook 2: Creating a Digital Schedule of Activities (USDM-HB2)** presents the study schedule of activities (SoA) as a USDM schedule timeline, and discusses its downstream automation (encounters, timings, activities, and the binding of activities to BCs; available at https://www.cdisc.org/USDM-HB2).
- **USDM Handbook 3: Generating Define-XML (USDM-HB3)** describes generating the Define-XML document from USDM metadata, consuming the BC bindings established in HB2 and the study identification and standards metadata established in HB1.

USDM-HB3 progresses from the earlier handbooks rather than duplicating them. In particular, the activity-to-BC assignments described in USDM-HB2 are the entry point for this handbook: They are what make the set of in-scope specializations, and hence the Define-XML content, derivable from the study definition.

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Scope and Boundaries

In scope

- Assembly of Define-XML metadata (study, standards, dataset, variable, value-level, and codelist metadata) from USDM metadata, SDTM DSS, and the CDISC Library
- Generation of a schema-valid Define-XML document from that assembled metadata
- Generation of a valid Define-XML document as a specification before data collection begins
- Reuse of the BCs and SDTM DSS referenced in the USDM metadata as building blocks for value-level metadata and controlled terminology

Out of scope

- Mapping specifications and source-to-target data transformations (how raw or collected data becomes SDTM)
- ADaM Define-XML generation
- Creation of SDTM datasets, or of analysis datasets.
- Table shells, statistical analysis outputs, and analysis results metadata; although Define-XML supports the latter, it is not derived from the USDM metadata here
- Front-end authoring interfaces for the study definition; this handbook assumes a USDM study definition already exists
- Updating the Define-XML at the conclusion of a study in preparation for a submission

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

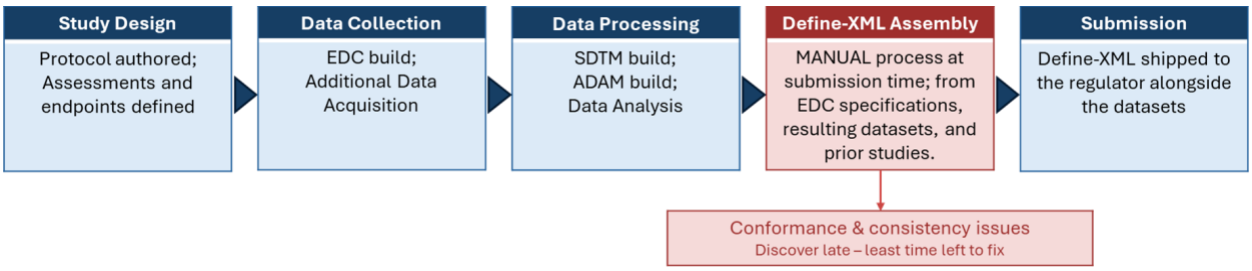
2 Relevant Workflows and Processes

2.1 Business Level

Define-XML (https://www.cdisc.org/standards/data-exchange/define-xml) provides the metadata describing the structure of each tabular dataset in a study. When used with the CDISC foundational standards, it provides metadata for human and animal model datasets using SDTM and/or the Standard for Exchange of Nonclinical Data (SEND), and for datasets using ADaM. The datasets in these formats and the corresponding Define-XML are required by the Food and Drug Administration (FDA; US) and the Pharmaceuticals and Medical Devices Agency (PMDA; Japan) for every study in each electronic submission. They inform regulators which datasets, variables, controlled terms, and other specified metadata were used to create the datasets.

The Business Problem

As illustrated in the following figure, in traditional workflows the Define-XML document is produced retrospectively, close to database lock and submission. When creating the Define-XML for SDTM datasets, a statistical programmer or standards specialist assembles it by inspecting the SDTM datasets, the electronic data capture (EDC) specifications, the corresponding annotated CRF, prior study Define-XML files, and the SDTM Implementation Guide (SDTMIG; https://www.cdisc.org/standards/foundational/sdtmig). These individuals often need to to re-key value-level metadata and codelist subsets that were, in principle, already decided when the study was designed. Because this assembly happens late and largely manually, any conformance and consistency problems (e.g., missing codelist subset, value-level condition that does not match data, an origin that was never specified) are discovered at the worst possible time, when there is the least time in the schedule left to correct them.



Value Proposition

Generating the Define-XML document automatically from the structured study definition stored in USDM format changes when and how metadata are produced. The structured study definition includes assessment details as biomedical concepts (BCs). Because BCs already carry the variable-level semantics, data types, and terminology subsets, much of the value-level and codelist metadata can be generated rather than transcribed. This enables the Define-XML to serve as a specification before data collection begins, rather than a retrospectively generated artifact created after the study ends. The benefits include

- earlier detection of metadata issues,
- a shorter and more predictable Define-XML production timeline, and
- consistent metadata across the stages at which a Define-XML may be needed (e.g., data safety monitoring board data cut, interim lock, unblinding, final submission).

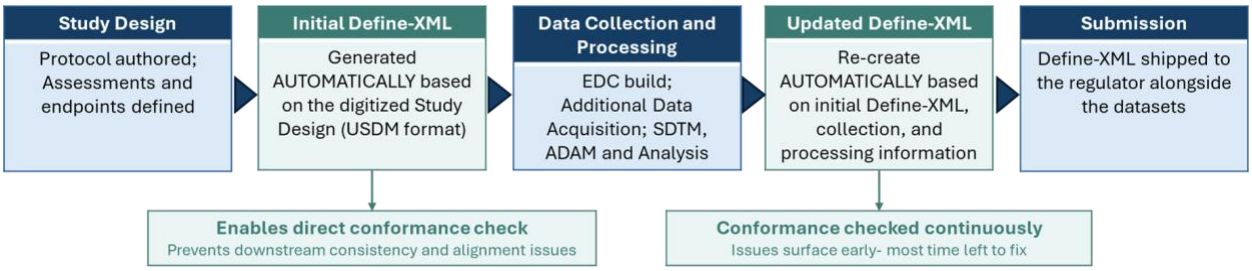
Strategic Impact

Beyond these immediate efficiency gains, USDM-driven Define-XML generation enables a digital-first submission process in which submission metadata is a by-product of study design rather than a separate downstream effort. It promotes reuse of CDISC standards content—the same BC feeds the CRF, the SDTM specification, and the Define-XML—and establishes the metadata foundation on which automation of the wider submission package can be built. It generates the Define-XML as soon as the study design is available, enabling it to serve as a study specification rather than merely reflecting a completed study.

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

End-to-end Digital Data Flow

In the anticipated future state, the study design is directly captured in the digitized USDM format. The USDM SoA includes references to the detailed assessments to be performed as BCs. The future solution reads the USDM file, selects the assessments specified for the study, and retrieves the associated SDTM DSS from the CDISC Library (https://www.cdisc.org/cdisc-library). These instantiations provide much of the detailed metadata needed to generate a Define-XML for use as a specification, including value-level metadata. In the future state, as illustrated in the following figure, the Define-XML is generated much earlier in the study process, enabling upfront conformance checking and early issue identification. Moreover, the Define-XML can be regenerated iteratively as the design evolves, rather than only once at submission. Each regeneration is inexpensive because it re-renders structured metadata rather than reassembling it by hand, and can inform on conformance and alignment issues.



2.2 User Level

The primary audience is the group responsible for turning a study definition into submission metadata: clinical/statistical programmers, data managers, standards leads, and the teams responsible for data collection infrastructure and study build.

- **Clinical/statistical programmers** need structured, complete variable and value-level metadata to generate a conformant Define-XML without transcribing it from datasets and prior studies. Currently, spreadsheet content is often used to represent the metadata needed to generate a Define-XML, and may need to be manually edited to provide the required metadata. At the start of the study, programmers cross-check the data collection instruments to ensure that everything needed for analysis is available. During SDTM implementation, they determine how to map the raw data to SDTM. At the end of the study, they update the Define-XML with any additional, unplanned metadata and generate the reviewer's guide. Updating the Define-XML file at the end of the study often occurs under tight deadlines.
- **Data managers** need the metadata used for data acquisition, reporting, and reconciliation to remain consistent with the metadata described to the reviewer. Any changes require updates to maintain alignment throughout the clinical study data pipeline.
- **Standards leads** need assurance that the Define-XML aligns with the applicable SDTMIG, CDASHIG, and CDISC Controlled Terminology versions, and that specialization content has been applied correctly.

Future End-to-end Digital Data Flow Phases with Key User Tasks

1. **Input preparation.** The key user task is to confirm the bindings of BCs (and, where relevant, the CDASH and SDTM DSS) to the study activities. The process starts from a USDM study definition in which activities are bound to BCs—and, where no published concept exists, to concept surrogates.
2. **Standards alignment.** Key user tasks include reviewing metadata completeness and supplying operational metadata that the USDM does not carry. The process involves resolving each referenced BC to its SDTM DSS and determining the in-scope SDTM domains, value-level metadata groups, and codelists. Specializations are located via their references in the USDM and retrieved from the CDISC Library (by identifier or by searching for concept grouping, definition, or synonyms when a reference is not present).
3. **Metadata assembly.** The key user task is to generate the initial Define-XML document from the metadata assembly. The metadata assembly combines SDTM domain and variable metadata (from the CDISC

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Library) with the specialization-derived value-level metadata and controlled terminology to populate the assembled Define-XML metadata.

4. **Metadata refinement**. The key user task is to iteratively refine the metadata to meet the study's needs until the Define-XML metadata is complete. This includes adding any missing study-level metadata (e.g., length or key sequence) as well as any variables or datasets needed for the study that were not included in the previous step. Missing metadata can be identified via placeholder values in the original Define-XML generated.
5. **XML generation.** Key user tasks include generating the Define-XML for each iteration, validating the output against the XML schema and the Define-XML conformance rules (https://www.cdisc.org/standards/foundational/define-xml), and generating the HTML visualization of the Define-XML document.

2.3 Technical Level

Core Concepts

The technical process rests on 7 building blocks:

- **USDM structures** — study version, study design, activities, and the BC references that bind meaning to those activities
- **Biomedical concepts** — standards-agnostic, semantically defined units of knowledge (e.g., systolic blood pressure) that fill metadata gaps by adding variable relationships and the detailed metadata needed to generate a CRF or a Define-XML document
- **CRF specializations** — the collection-oriented implementation layer of BCs for Clinical Data Acquisition Standards Harmonization (CDASH), a ready-to-use building block for generating CRFs; in Define-XML, a dataset variable must trace back to how it was collected
- **SDTM DSS** — the implementation layer of BCs for SDTM which includes ready-to-use building blocks and provides value-level metadata for assessments directly related to a domain (e.g., the systolic blood pressure specialization in the Vital Signs domain)
- **SDTM dataset specification** — the specifications of each domain as specified in the SDTMIG including detailed metadata from the CDISC Library; this extends beyond content available in the DSS
- **SDTM controlled terminology** — the terminology required to generate codelist metadata for submissions
- **Define-XML** — the target artifact, a machine-readable description of datasets, variables, value-level metadata, codelists, and documentation

The Metadata Model (What Flows, and What Is Missing)

The USDM, its referenced BCs, and SDTM DSS provide much but not all the metadata needed to generate a complete Define-XML. These digital sources provide the set of in-scope domains and variables (from the activities and their specializations), variable data types, controlled terminology and subset codelists, variable roles, and the conditions that define value-level metadata. Additional metadata is retrieved from the SDTMIG in the CDISC Library, including the full SDTM dataset-level metadata (class, structure) and the complete standard variable list for each domain. Some study-specific metadata (e.g., variable lengths, significant digits, display formats, key sequences, derivations, comments, document references) are not available via CDISC-provided sources. This missing study-specific metadata is added to the Define-XML using a refinement process. Upon completion of the refinement process, the Define-XML has a complete, valid set of metadata.

The Optional Operational Specification Layer

Some implementations introduce an intermediate operational specification layer between the USDM and the Define-XML. Its role is to hold all the operational metadata needed to generate the Define-XML and other artifacts, based on metadata derived from the USDM and additional metadata from other sources. The operational specification metadata enables the generation of Define-XML from a single consolidated specification. In the CDISC 360i project, this operational layer is exemplified by the DDS.

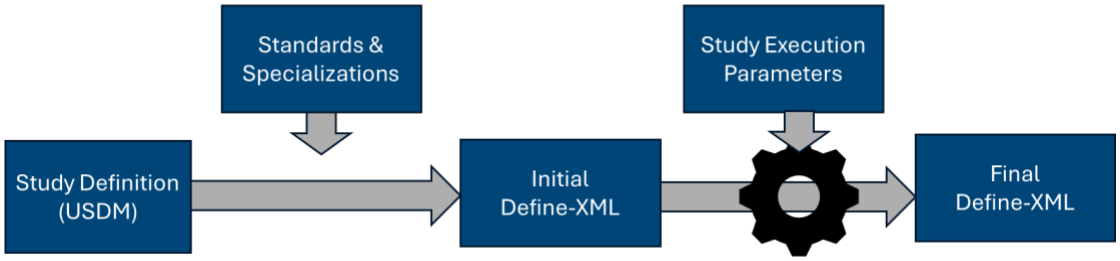
© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Implementation Options

The following figures illustrate different implementation options for using USDM to generate the final Define-XML. The study-specific design information stored in USDM format needs to be further enriched with operational information. This can be a direct mapping from all applicable sources (Option A), an intermediate specification layer (Option B), or a repository-mediated path (Option C).

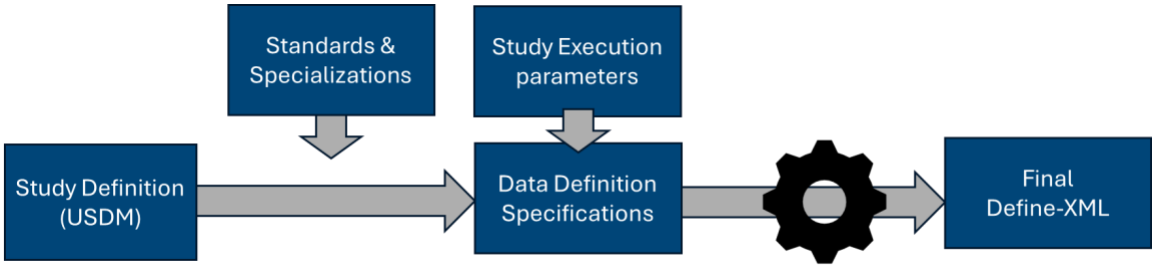
Option A — Direct Mapping

Use the USDM BC configuration to generate the Define-XML supported by the corresponding standards and BC specializations. Missing elements can be supplied inline and/or required during run-time in a dedicated solution. Iterate based on study execution parameters to create the final Define-XML for submission. This represents the lowest infrastructure level, which is best where operational metadata is simple or standardized.



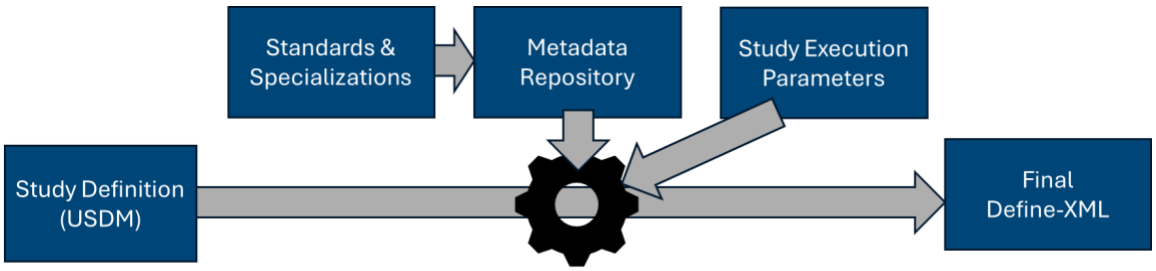
Option B — Driven by Data Definition Specifications (DDS)

An operational specification layer (the DDS) sits between the USDM and the Define-XML, capturing operational metadata that the USDM does not carry before the document is generated. This consolidated operational specification layer is enriched with methods, comments, and document references. The Define-XML can be generated from it. This option is better for reuse and governance across studies.



Option C — Repository-mediated

A solution drives generation through a sponsor metadata repository (MDR) or other systems within the sponsor's information technology infrastructure that hold company-standard defaults, prior-study metadata, and operational metadata. The USDM is used to scope and seed the content.



© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Versioning and Standards Alignment

A generated Define-XML must identify and remain internally consistent with the specific standard versions in play: the Define-XML version as the target format, the applicable SDTMIG version, the CDASHIG version where collection traceability is represented, and the CDISC Controlled Terminology package. The SDTM DSS declares the SDTMIG version range to which they apply, and the CT version is fixed by the study's standards metadata; both must be reconciled during assembly.

2.4 Maturity Matrix

Organizations adopt USDM-driven Define-XML generation incrementally rather than in a single leap, and it must be noted that value accrues well before the highest level: even template-based generation removes most of the transcription risk that characterizes the manual process.

The five levels described in the following table represent a path rather than a prescription. At level 1, the Define-XML is assembled by hand; introducing company templates and codelist libraries reaches a second level (semi-automation). The pivotal step is the third level—this is where the study definition, rather than a person, determines the content of the Define-XML. Beyond it, level 4 adds an operational specification layer and a repeatable pipeline in, and in level 5 the Define-XML is automatically generated and validated.

Level	Description	Characteristics
Level 1	Manual Define-XML creation	Define-XML assembled by hand from datasets, aCRF, and prior studies; no reuse of structured concept metadata
Level 2	Semi-automated using templates	Company Define-XML templates and codelist libraries reused; value-level metadata still entered manually
Level 3	USDM-driven metadata generation	In-scope specializations, value-level metadata, and codelists derived from the BCs referenced in the USDM
Level 4	Metadata-driven pipeline	USDM plus an operational specification layer feed a repeatable pipeline; operational metadata managed centrally and reused across studies
Level 5	Fully automated digital pipeline	Define-XML generated and validated automatically from the study definition at any milestone, with human review at defined checkpoints

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

3 Solution Requirements

This section describes what a solution must do to generate a Define-XML file from the USDM, and the qualities it must exhibit to be trusted in a regulated setting. These topics are intended as a starting point for a fit-for-purpose assessment: An organization can weigh each requirement against its own maturity, tooling, and standard operating procedures. The requirements are grouped into stakeholders with an interest in the outcome; the functional capabilities the solution must provide; the non-functional qualities it must exhibit; and the system-level inputs, outputs, and interfaces it must support.

3.1 Relevant Stakeholders

Define-XML generation touches almost every function that handles study metadata, and each stakeholder approaches it from a different viewpoint.

Internal stakeholders are the study team; **external stakeholders** sit at the boundary of the organization, but nonetheless shape or consume the result. What unites them is a shared interest in a single, consistent source of metadata—the value of USDM-driven generation is realized only when these groups treat the structured study definition, rather than their own local copies, as the authoritative source.

Within the study team, data management is concerned that the metadata used to acquire, report, and reconcile data stays consistent with the metadata ultimately described to the reviewer; a divergence between what was collected and what is documented is exactly the kind of late finding the approach is meant to prevent. Clinical and statistical programmers are the primary producers of the Define-XML and benefit most directly, because complete, structured variable and value-level metadata remove the transcription effort that dominates manual assembly. Medical writers aim to keep the protocol and the structured study definition aligned, since both flow from the same design decisions. Standards governance is accountable for correct alignment with the applicable SDTMIG, CDASHIG, and Controlled Terminology versions, and for controlling how specializations and codelist subsets are reused across studies.

Among external stakeholders, the regulatory agency is the ultimate consumer: It requires a valid, conformant, navigable Define-XML, and it is the reason the document exists at all. System developers need a clear, stable mapping from the USDM to Define-XML, along with reference examples to guide development and improve interoperability. Contract research organizations (CROs), which frequently perform study build and submission work on a sponsor's behalf, need metadata that is interoperable and reusable, and which travels with the study definition rather than living in a proprietary format. The following table summarizes each group's vantage point and principal interests.

Role	Responsibility	Viewpoint	Type	High-level interests
Management	Overall delivery of the study; investment decisions on tooling and process change	Strategic	Internal	Reduced cycle times and rework Compliance and inspection readiness Return on tooling investment
Clinical & Statistical programmers	Creation of Define-XML	Analysis and reporting	Internal	Complete, structured metadata to generate conformant Define-XML efficiently. Easy, intuitive tooling for review, adding information, and creation of Define-XML
Data manager	Creation and/or retrieval of additional metadata needed for the creation of Define-XML.	Data capture and standardization	Internal	Consistency of metadata across acquisition, reporting, and reconciliation. Direct automation from EDC to Define-XML for informing metadata Easy, intuitive tooling for review and adding metadata
Medical writer	Study design representation	Study design documentation	Internal	Correct and consistent representation of study designs in submission metadata Seamless study design features
Standards lead	Ensure alignment with guidance and standard terminology	Requirements and output	Internal	Correct alignment with SDTMIG, CDASHIG, and Controlled Terminology; controlled reuse Easy, intuitive tooling that enables the creation of Define-XML based on the specified IGs
Quality Control	Validation of output against requirements	Requirements and output	Internal	Correct alignment with SDTMIG, CDASHIG, and Controlled Terminology; controlled reuse

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Role	Responsibility	Viewpoint	Type	High-level interests
				Correct representation of the information that is (to be) collected for a specific study Easy, intuitive tooling that enables review of the requirements represented in the IGs with the actual metadata created for the study
Regulatory agencies	Review of submitted study information	Reviewer	External	Valid, conformant, navigable submission metadata (the submission target)
System developers	Implementation and integration of the solutions	Implementation	External	Input and output requirements for the solution Technical implementation details Clear USDM-to-Define-XML mapping and reference examples to build against.
CROs	Study set-up and execution	Consumer	External	Interoperable, reusable metadata that travels with the study definition Automated set-up of processes and systems based on the metadata provided

3.2 Upstream Requirements for the Study Design Solution

The study design solution (e.g., a study design software application) must create the study design in USDM format. Flexible, intuitive tooling and features will ensure correct, usable content that works downstream. To create the Define-XML document, the SoAs and corresponding assessments (i.e., BCs) must be stored in USDM format. For a full description of the corresponding requirements, see USDM-HB2.

3.3 Downstream Requirements for the Creation of Define-XML

The described solution generates a Define-XML file (define.xml) based on the study design. This file will be shared with data managers and clinical/statistical programmers to function as a specification for the SDTM dataset generation process. A complete Define-XML file contains the metadata needed to support automated SDTM generation processes and to validate that SDTM datasets conform to the specification. As the study design is updated, the Define-XML file will be regenerated to create an up-to-date specification.

At the conclusion of the study, the Define-XML file may be refined to accurately reflect the study content for a regulatory submission. Generating and applying a Define-XML file as a specification based on the study design minimizes the effort required to prepare it for inclusion in a submission.

3.3.1 Functional Requirements

The solution needs to identify the information and structure required for the Define-XML document, as described in the CDISC documentation (https://www.cdisc.org/standards/data-exchange/define-xml). Based on this, the solution must be able to generate the key Define-XML elements from structured inputs alone, without requiring the metadata to be re-entered by hand. The inputs include

- the USDM study definition,
- the SDTM DSS that are referred to and/or matched, and
- the SDTMIG dataset metadata and CT available in the CDISC Library.

Using the domain metadata, the solution identifies and adds any expected or required variables not referenced in the SDTM DSS. Furthermore, the solution needs to identify any missing metadata and provide the means to add it. Ideally, automated methods add the study-specific metadata needed to complete the Define-XML document, but in some solutions manual entry may be required.

The solution must also enable metadata reuse, so that content proven in one study can be carried forward to the next. The solution must validate its output, confirming that the document conforms to the Define-XML schema and conformance rules for the indicated version and satisfies the published conformance rules.

The solution needs to preserve traceability: Every element of the generated Define-XML should be relatable back to the activity, BC, and specialization from which it was derived, so that a reviewer or auditor can follow the lineage

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

from the study design through to the submitted metadata. Traceability is what turns generation from a convenience into an assurance. It lets the organization demonstrate, rather than assert, that the metadata describe the study as designed.

3.3.2 Non-functional Requirements

Beyond these capabilities, the solution must exhibit the qualities expected of any process that contributes to a regulatory submission. It must be auditable: for any generated Define-XML file, it should be possible to recover which study version, specialization package, and CT package the document was built from, so the result can be reconstructed and explained after the fact. It must perform well enough to be run iteratively. The central premise is that a Define-XML file can be produced early and regenerated as the design evolves, which is practical only if generation is fast and inexpensive rather than a multiday undertaking. It must also be reproducible: Given the same inputs, the solution must produce the same output, so any difference between 2 Define-XML documents can be attributed to a genuine change in the study rather than to the generation process itself.

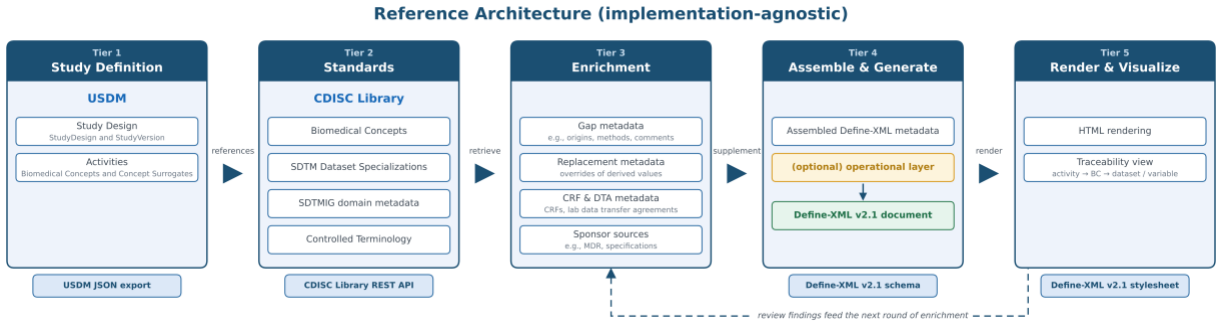
3.3.3 System Context and Architecture

At the system level, the requirements resolve into the inputs the solution must accept, the outputs it must produce, and the interfaces through which it must exchange them. These are summarized in the following table, and are deliberately expressed in terms of open standards so that any conformant component—a study design solution, a metadata repository, a scripting environment, or a commercial toolkit—can satisfy them.

Category	Requirement
Input	• A structured USDM study definition including BC instantiations with SDTM DSS references, where available • SDTM DSS • SDTM domain metadata and CT available in the CDISC Library and/or company MDR • Additional study-specific metadata needed to complete the full Define-XML file
Output	A schema-valid Define-XML document (define.xml file) that passes the conformance rules with consistent dataset, variable, value-level, and codelist metadata, plus a human-readable rendition
Interoperability	Ability to integrate with EDC systems, metadata repositories, and submission systems, and to consume the CDISC Library API for standards and terminology.

Reference Architecture

The reference architecture has 5 tiers, as presented in the following figure. The figure denotes roles in the flow, not specific products.



Tier 1 includes the study definition, represented by the USDM API file as a JSON export. Each relevant activity for the study represented in this file references 1 or more BCs or BC surrogate.

Tier 2 is the standards tier which can be retrieved from the CDISC Library. This includes the BC definitions, the SDTM DSS that implement them for the purpose of creating SDTM domains, the SDTMIG domain and variable metadata, and the SDTM CT.

Tier 3 is the enrichment tier. It iteratively adds the additional metadata needed to address metadata gaps and study-specific requirements not covered by tier 2.

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

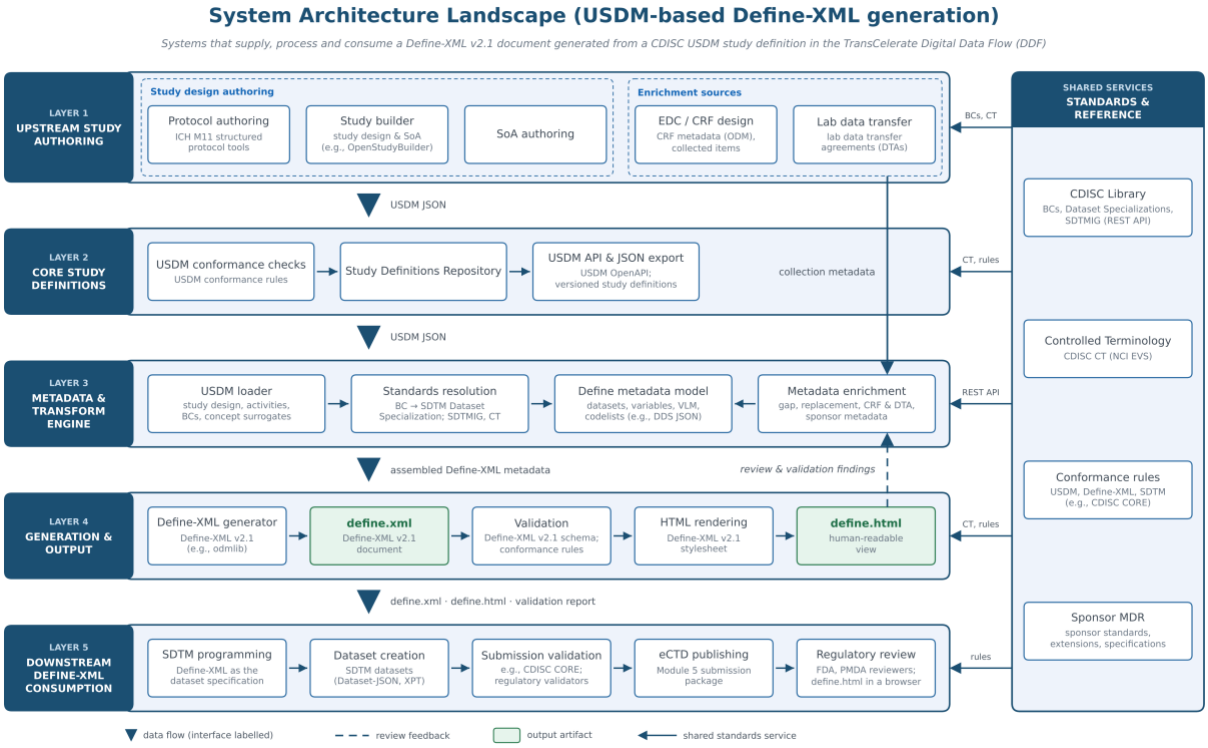
Tier 4, the assembly and generation tier, combines all inputs into Define-XML metadata structures and generates the Define-XML document.

Tier 5 is the render-and-visualize tier, which displays the Define-XML so users can interpret and review it. Findings based on the review can feed into the enrichment tier, improving the correctness and completeness of the Define-XML output.

Each tier can be filled by any conformant system—an MDR, a study design solution, a scripting environment, or a commercial toolkit. The interfaces between tiers (the USDM export, the CDISC Library API, and the Define-XML document) are open standards. This handbook prescribes what each tier contributes, not the technology that provides it. Note that CDISC membership is required to use the CDISC Library API.

System Architecture Landscape

Whereas the reference architecture landscape shows many of the systems typically involved in generating Define-XML from a USDM study definition, the system architecture describes how the metadata move between tiers. The following figure shows the systems that supply that metadata, process it, and consume the resulting Define-XML. Systems are named by role; products in parentheses are examples, not requirements. The dashed arrow shows that generation is iterative: Findings from validation and from review of the rendered define.html feed the next round of enrichment. Each round moves the Define-XML from an initial specification toward a submission-ready document. The side rail shows shared standards and reference services that support every layer.



The system architecture landscape is organized into 5 layers.

- **Layer 1** includes the protocol authoring, study builder, and SoA tools to create the study design and write it as USDM JSON. A second group of enrichment sources—EDC/CRF design and lab DTAs—supplies collection and transfer metadata that the study design does not carry.
- In **Layer 2**, the core study definitions are checked against USDM conformance rules and a study definitions repository—either the Digital Data Flow Study Definitions Repository (DDF SDR; see https://github.com/transcelerate/ddf) or a sponsor equivalent—which provides versioned study definitions through the USDM API and JSON export.

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

- In **Layer 3**, the engine loads the USDM study design and resolves each BC to its SDTM DSS, SDTMIG domain metadata, and CT. From these, it builds an intermediate Define-XML metadata model. Metadata enrichment then adds the gap, replacement, CRF, DTA, and sponsor metadata that the primary sources cannot provide.
- The Define-XML document generated in **Layer 4** is validated against the Define-XML schema and conformance rules, then rendered to HTML with the Define-XML stylesheet.
- In **Layer 5**, Define-XML serves as the dataset specification for SDTM programming and dataset creation (e.g., Dataset-JSON, SAS XPT). It accompanies the datasets through submission validation, eCTD publishing, and regulatory review.

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

4 Technical Implementation

This section describes how a Define-XML document is assembled and generated from the USDM. It first presents the reference architecture, then walks through the metadata chain from a USDM activity to a value-level metadata entry; gives the element-level mapping; addresses the study-specific metadata that none of the standard sources provide; and, finally, discusses implementation options, tooling, and the relationship to the process described for SDTM DSS in prior work.[1]

4.1 From USDM to Define-XML: The Metadata Chain

The USDM makes Define-XML content derivable because it records, for each activity, which BCs are observed, and—through study-specific configuration and/or extension references—which SDTM DSS implements each concept.

The metadata chain that explains how the USDM makes Define-XML content derivable traces back to the System Context and Architecture. Step 1 in the metadata chain maps to tier 1 in the Reference Architecture and extends into the BC and SDTM DSS in tier 2. Step 1 maps to layers 1-3 in the System Architecture Landscape, with layer 3 partially covered to include the loading of a USDM version and matching of various input standards (i.e., standards resolution). Steps 2-4 in the metadata chain map to tier 2 in the Reference Architecture and layer 3 in the System Architecture Landscape. Steps 5-6 map to tier 4 in the Reference Architecture and layer 4 in the System Architecture Landscape.

Step 1: Input Preparation – USDM Binding of Activities

In the USDM, the study version contains a study design whose activities represent the study assessments and procedures. Each relevant activity references its configured BCs via biomedicalConceptIds.

Each BC in the USDM carries

1. a reference to the CDISC Library or internal MDR BC code;
2. an optional extension attribute pointing to its SDTM DSS;
3. a set of properties that correspond to the SDTM variables represented in the SDTM DSS, each with a data type, requiredness and enablement flag, and a standard BC data element NCI code; and, where applicable,
4. response codes, including an enablement flag for the study.

The following example shows the reference to the configured BC in a sponsor MDR library and a reference to a DSS via the extension attribute mechanism in USDM.

```
"biomedicalConcepts": [
          {
            "id": "BiomedicalConcept_2",
            "name": "ICF",
            "label": "Informed Consent",
            "synonyms": [
              "Informed Consent Obtained"
            ],
            "reference": "//sponsor_mdr//ConfiguredBCs/v1/ICF1",
            "extensionAttributes": [
                {
                    "id": "ExtensionAttributeValue_1",
                    "url": "//ExtensionLibrary/USDMv4_0/SDTMSpecialization/v1/reference",
                    "valueString": "/mdr/specializations/sdtm/packages/2025-12-
16/datasetspecializations/CONSENT",
                    "extensionAttributes": [],
                    "instanceType": "ExtensionAttribute"
                }],
            "properties":
            ....
```

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

As described in USDM-HB2, the data element concepts from the BC library are stored in USDM as properties. These properties, along with their corresponding expected response codes, indicate what information to collect for the study. This can be

- a fixed value (e.g., informed consent protocol milestone) and corresponding category (see Appendix A1, Instantiated BC: Informed Consent);
- a set of optional responses of which only 1 is allowed (e.g., for body weight units where only kilogram is enabled; see Appendix A3, Instantiated BC: Weight); or
- a full options codelist, from which 1 has to be chosen during study execution (see fasting example in Appendix A2, Instantiated BC: Platelet Count).

Full USDM examples are available in the USDM DDF-RA repository (https://github.com/cdisc-org/DDF-RA/tree/main/Documents/Examples), updated with each version of the USDM.

This USDM binding, established as described in the USDM-HB2 study-design workflow, is the entry point for Define-XML generation. SDTM DSS corresponding to the configured BC can be identified either by matching the BC NCI code and enabled responses or via a USDM extension that points to the specialization. In this example, the specialization reference is carried as a USDM extension attribute, and the corresponding DSS vlm_group_id. The library of current BC SDTM specializations is available on the CDISC website (https://www.cdisc.org/cdisc-biomedical-concepts).

Step 2: Identifying Define-XML Domains and Variables Specified in the USDM

Variables and Domains Identified by BCs and Specializations

For each referenced BC, the corresponding SDTM DSS is to be retrieved from the CDISC Library—directly by the identifier carried in the USDM extension, or—where a reference is absent—by searching the Library for a match on the concept code and property definitions like labels, isEnabled properties, and response codes. In cases where the BC has more than 1 corresponding DDS, the Define-XML generator should provide a mechanism to select the correct DSS.

Retrieval uses the CDISC Library REST API; the relevant endpoints are summarized in the following table (illustrative; an API key is required). See the CDISC Library Service Desk Knowledge Base (https://wiki.cdisc.org/spaces/LIBSUPRT/overview) for technical details.

Endpoint	Returns
/mdr/bc/packages	List of BC packages
/mdr/bc/packages/{package}/biomedicalconcepts/{bc}	BC definitions
/mdr/specializations/sdtm/packages	List of SDTM DSS packages
/mdr/specializations/sdtm/packages/{package}/datasetspecializations/{id}	An SDTM DSS (domain, variables, VLM)
/mdr/products/terminology/packages/.../codelists/{id}	A CT codelist and its terms

The DSS includes the SDTM domain and variable information. The union of the USDM with DSS determines which SDTM domains and variables appear in the study-specific Define-XML. This may also link to the BC CRF specializations, which provide collection details and inform setup of the corresponding EDC system.

The union of USDM with DSS tells the Define-XML generator which building block to use; the BC properties specified in USDM tell it what to instantiate from the building blocks specified in the SDTM DSS. Each USDM property corresponds to an SDTM specialization variable and carries 2 flags that govern its contribution to the Define-XML.

1. The isEnabled flag indicates whether the property participates in this study's use of the concept: only enabled properties are instantiated; properties defined with "isEnabled = false" are omitted. The set of enabled properties determines which of the specialization's variables and value-level metadata targets appear for this study. For example, VSLAT is part of the BC, but may have "isEnabled = false" in the USDM file if that variable will not be collected or included in the final Vital Signs dataset.
2. The isRequired flag indicates whether the property is mandatory for the concept as used in the study.

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Although a property might not be enabled or included in the USDM, it still might need to be included in the Define-XML. The following table outlines the logic for including properties specified in the USDM and/or the SDTM DSS in Define-XML. Ultimately, both the SDTMIG requirements (see step 3) and whether data is collected must be evaluated to ensure the Define-XML is complete.

USDM property	USDM property code	Matching to SDTM BC specialization	In Define-XML?
Enabled	CDISC C-code	Available based on data element C-code or if not available, based on property label or alternative[a] or if no match, check with SDTMIG[b]	Included
Enabled	Extension code used or not available	Based on property label or alternative[a] or if no match, check with SDTMIG[b]	Included
Not Enabled	CDISC C-code	Available based on data element C-code or if not available, based on property label or alternative[a] or if no match, check with SDTMIG[b]	Included if collected and/or mandated in SDTMIG
Not Enabled	Extension code used or not available	Based on property label or alternative[a] or if no match, check with SDTMIG[b]	Included if collected and/or mandated in SDTMIG
Not Included		Available in SDTM BC specializations	Included if collected and/or mandated in SDTMIG

[a]Although not required, it is helpful when the property label contains the corresponding SDTM variable name when it is not included in the standard BC library. However, other methods might become available in the future.

[b]See Verification with the SDTM Domain Definition.

Verification with the SDTM Domain Definition

The SDTM domain matching the specified BC is indicated in the SDTM BC specializations (DSS domain attribute). The unique set of domains identified by this method needs to be included in the Define-XML. The corresponding domain requirements for submission are further detailed in the SDTMIG. Matched enabled variables are treated as needed for the study and emitted as a mandatory item (whereas a permissible variable whose property is "isEnabled = false" is dropped from the domain's variable list for the study). In the unlikely event that a property is not required (isRequired=false) but is enabled (isEnabled=true), and the information is collected for the specific study, the corresponding variable should be added to the Define-XML definition as well.

In effect, the flags turn the open-ended set of permissible variables into a determinate, study-specific selection: isRequired promotes the permissible variables the study relies on, and disabling removes the permissible variables it does not use; required and expected variables are unaffected because their presence is mandated by the standard rather than by the concept. This is what allows a generated Define-XML to describe exactly the variables the study uses, rather than the full permissible superset of all domains. The following table summarizes how the SDTMIG core designation and the property flags are combined to determine whether a variable is included.

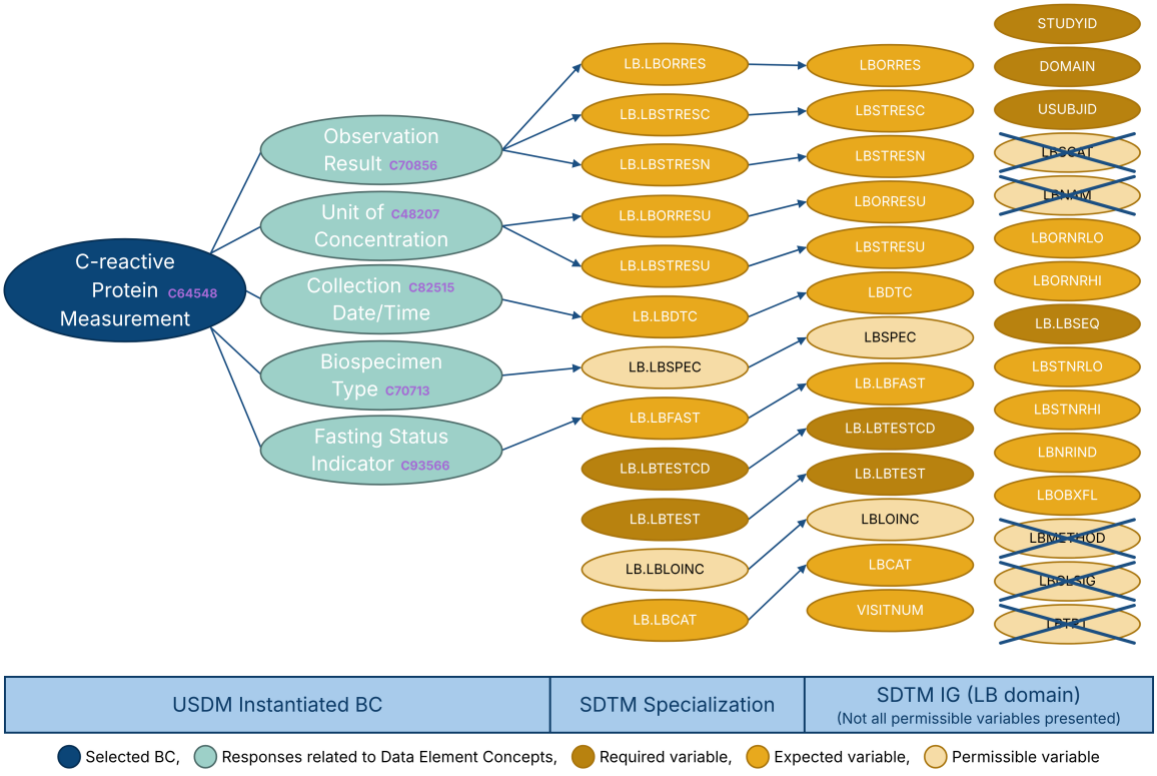
SDTMIG core	BC property flag	In Define-XML?	Effect on the variable
Required	any (or none)	Included	Mandated by the SDTMIG; always present, emitted as Mandatory = Yes
Expected	any (or none)	Included	Mandated by the SDTMIG; present even when null
Permissible	isRequired = true	Included (mandatory)	Promoted by the concept: selected and emitted as a mandatory item
Permissible	isEnabled = true (not required)	Included (optional)	Selected because the concept uses it; emitted as a non-mandatory item
Permissible	isEnabled = false	Excluded	Dropped from the study's variable list for this domain
Permissible	no property present	Excluded	Not referenced by any in-scope concept; omitted for the study

Note: Required and expected variables are shown for completeness; flags apply only to permissible variables, where the study-specific selection is made.

The following figure shows an example of how a concept selected from the BC library links to the corresponding specializations and to the information in the SDTMIG. The SDTMIG includes a superset of variables compared to the original BC and the SDTM DSS. Variables identified as permissible in the SDTMIG that are not included in the USDM BC specification should be removed from the domain specification in Define-XML.

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Note that multiple BCs specified in a USDM-formatted study definition may contribute to the same SDTM domain. The unique set of all requirements (either by enabled=true in USDM, or the required or expected setting in SDTMIG) should be included for the domain in Define-XML. So, if any of a set of BCs contributing to the same domain enable a permissible variable, then it needs to be included it in the final Define-XML specification for that domain regardless of whether it is enabled for the other contributing BCs.



As indicated in USDM-HB2, specifying additional SDTM and EDC variables in USDM as properties helps downstream automation and interoperability between collection and submission information. A study refinement process may add additional variables or datasets to support the study's needs, even when those variables are not indicated in the original foundational BC and/or SDTMIG. These additional variables also need to be included in the generation of a Define-XML file.

The LOINC code is a special case in the USDM-to-SDTM transition process. USDM allows many alternative codes, including LOINC codes for BCs. This allows for direct mapping and downstream interoperability. The LOINC code is not stored as a property itself but is directly linked to the BC C-code. If LOINC codes are specified, they need to be recognized by the corresponding codeSystem attribute in the Define-XML generator and can then be copied to the LBLOINC variable in SDTM. See Appendix A2, Instantiated BC: Platelet Count, for an example of alias code usage of the LOINC code in USDM.

Timing Variables and Subject Visits Domain

Although specified in the SDTMIG SV domain specification, timing variables for visits and more granular planned timepoints (e.g., VISITNUM, VISIT, --DY, --TPT) are typically not included in BC definitions. However, the USDM does inform the need for adding these variables based on its timeline capabilities. The following logic can be applied to that:

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

- If the variable is designated as required or expected in the SDTMIG, it must be included.
- If the assessment is performed on multiple days during the study, then the VISIT, VISITNUM, and __DY variables are likely to be included.
- If the assessment is performed at multiple timepoints during a day within the study, then the __TPT and corresponding variables should be included.

The timeline stored in USDM indicates when an assessment needs to be performed and points to the activity that refers to the corresponding BCs. Each reference from a timepoint instance in the timeline via the activity to the BC indicates that the assessment needs to be done at that planned time. The same timepoint will also refer directly or indirectly (via a subtimeline) to the corresponding visit definition, which is further detailed in the Trial Visits domain. (The creation of this domain and the corresponding Define-XML input is described in USDM-HB1.) The corresponding SV domain can be defined based upon that and the corresponding requirements in the SDTMIG. In case multiple timings are defined for a visit, either in the main or subtimelines (see USDM-HB2), then corresponding timepoint details (e.g., --STRTPT, --STTPT, --ENRTPT, --ENTPT) also need to be added. In most cases additional reference timepoint variables (e.g., --TPTREF, --RFTDTC) might be needed.

Note that if more BCs are specified for the same domain (e.g., Vital Signs, Laboratory Test Results), then the unique set of all timings for the corresponding BCs needs to be specified in Define-XML for the corresponding domain.

Step 3: Identifying Permissible Values

Where a property has a controlled set of permissible values, those values are carried as responseCodes in USDM. Each response code pairs a submission value with a CDISC/NCI code and decode, along with the code system and its version. The units property VSORRESU of the Systolic Blood Pressure concept, for example, carries a single response code for mmHg; the property's own code is the Data Element concept it represents (Unit of Pressure), while the response code is the value the variable may take:

```
{
  "name": "VSORRESU",
  "isRequired": false,
  "isEnabled": true,
  "datatype": "text",
  "responseCodes": [
    { "name": "mmHg", "isEnabled": true,
      "code": { "code": "C49670", "decode": "mmHg",
                "codeSystem": "http://www.cdisc.org",
                "codeSystemVersion": "2026-05-26" } }
  ],
  "code": { "standardCode": {
      "code": "C49669", "decode": "Unit of Pressure" } }
}
```

A response code can equally represent an enumerated result term rather than a unit. The DSDECOD property of the Informed Consent concept is required and carries the response code INFORMED CONSENT OBTAINED, anchored to NCI code C16735:

```
{
  "name": "DSDECOD",
  "isRequired": true,
  "isEnabled": true,
  "datatype": "text",
  "responseCodes": [
    { "name": "INFORMED CONSENT OBTAINED", "isEnabled": true,
      "code": { "code": "C16735",
                "decode": "INFORMED CONSENT OBTAINED" } }
  ]
}
```

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

These response codes enable the codelist metadata to be built without manual entry (see step 5). Each enabled response code becomes 1 item of the codelist for its variable: The response code's decode becomes the coded (submission) value, its label becomes the human-readable presentation value for use in documents and reports, and its NCI code and code system anchor the term in CDISC CT. A property whose enabled response codes are a subset of the terms in the referenced CT codelist yields a subset codelist—only the study's permissible values are emitted—whereas a property that carries a single fixed-response code contributes an assigned value rather than a choice. Together, the enabled and required flags and the response codes make the USDM BC a study-specific configuration of the specialization rather than merely a pointer to it.

The following example Define-XML codelist is based on the LBTESTCD and LBTEST properties defined in the USDM (this includes the example in Appendix A2). These properties, all with 1 single response value, are derived from multiple BCs and combined into a single codelist for the corresponding complete LB domain.

```
<CodeList OID="CL.LBTESTCD" Name="Laboratory Test Code" DataType="text"
def:StandardOID="STD.3" SASFormatName="$LBTESTC">
        <CodeListItem CodedValue="BILI">
          <Decode>
            <TranslatedText xml:lang="en">Bilirubin</TranslatedText>
          </Decode>
          <Alias Context="nci:ExtCodeID" Name="C38037"/>
        </CodeListItem>
        <CodeListItem CodedValue="GLUC">
          <Decode>
            <TranslatedText xml:lang="en">Glucose</TranslatedText>
          </Decode>
          <Alias Context="nci:ExtCodeID" Name="C41376"/>
        </CodeListItem>
        <CodeListItem CodedValue="PLAT">
          <Decode>
            <TranslatedText xml:lang="en">Platelets</TranslatedText>
          </Decode>
          <Alias Context="nci:ExtCodeID" Name="C51951"/>
        </CodeListItem>
        <CodeListItem CodedValue="HGB">
          <Decode>
            <TranslatedText xml:lang="en">Hemoglobin</TranslatedText>
          </Decode>
          <Alias Context="nci:ExtCodeID" Name="C64848"/>
        </CodeListItem>
        <Alias Context="nci:ExtCodeID" Name="C65047"/>
      </CodeList>
```

Step 4: Adding Value-level Metadata

A Reference Process for Value-level Metadata

A concrete, repeatable process for transforming SDTM DSS and CT into Define-XML value-level metadata has been established.[1] The process is described here in implementation-neutral terms: Each step is a metadata transformation with well-defined inputs and outputs, and it can be performed in any language or on any platform; Jansen[1] described these steps in SAS and the CDISC 360i Data Definition Engine (https://github.com/cdisc-org/data-definition-engine) generates a Define-XML v2.1 file, including value-level metadata, using Python. The process for generating a Define-XML described in this handbook can be implemented in any technology.

1. **Start from basic domain metadata.** Begin with study, standards, dataset, and variable metadata covering only the in-scope domains. In the USDM-driven flow, in-scope domains follow from the specializations referenced by the study's activities rather than being chosen by hand.
2. **Retrieve the SDTM DSS.** Obtain the applicable specializations from the CDISC Library (by the identifiers carried in the USDM or by searching) and consolidate them into a single table of specialization variables and their attributes.

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

3. **Retrieve the controlled terminology.** Obtain the SDTM CT package identified by the study's standards metadata from the CDISC Library. In 360i, for example, the CT package is provided as a command-line argument.
4. **Build the value-level metadata.** From the specialization variables, create the value-level metadata: variables with comparator EQ or IN become WhereClause conditions, and variables with vlmTarget = true become the described variables, with references to both full codelists and subset codelists.
5. **Build the codelist metadata.** Retain only those codelists referenced by the value-level metadata, and construct subset codelists from the specialization value lists.
6. **Generate and validate.** Serialize the assembled metadata to Define-XML, produce the HTML rendition, and validate against the XML schema and the Define-XML conformance rules.

The choice of language, database, or toolkit for these steps is an implementation detail; this handbook prescribes the metadata and the sequence of transformations, not the technology used to carry them out.

Generating Value-level Metadata Using SDTM Dataset Specializations

An SDTM DSS provides the metadata needed for value-level metadata. At the group level, it declares its datasetSpecializationId, domain, source, shortName, the SDTMIG version range it applies to, and the BC it implements. At the variable level, each variable carries the metadata that Define-XML value-level metadata requires:

Specialization variable attribute	Role in Define-XML
comparator (EQ / IN)	Marks variables that become conditions in the (composite) WhereClause of a value-level metadata group. For example: VSTESTCD EQ "SYSBP" AND VSPOS IN ("PRONE","SEMI-RECUMBENT", "SITTING", "STANDING", "SUPINE") AND VSLOC IN ("BRACHIAL ARTERY", "CAROTID ARTERY", "DORSALIS PEDIS ARTERY", "FEMORAL ARTERY", "FINGER", "RADIAL ARTERY") AND VSLAT IN ("LEFT", "RIGHT")
vlmTarget (true/false)	Marks the target variable(s) described by value-level metadata under those conditions
codelist / subsetCodelist / valueList	Supply the controlled terminology reference, a subset codelist, or an explicit value list for the variable
assignedTerm	Supplies an assigned (default) value for a condition or target variable
dataType, length, significantDigits, format	Populate the corresponding variable definition attributes
originType, originSource	Populate the variable Origin (e.g., Collected/Investigator, Derived, Assigned, Predecessor, Protocol)
role, relationship	Populate the SDTM variable role and record variable relationships (e.g., HAS_UNITS)

Selection and population are thus 2 distinct acts. Matching the USDM BC, either by extension reference or by property matching, selects the specialization; the BC properties described in step 2 populate it. The specialization retrieved from the CDISC Library provides value-level metadata, including the WhereClause conditions (comparator EQ/IN) and the described targets (vlmTarget), along with their codelist and subsetCodelist references. While the enabled properties fix the in-scope variables and targets, the required flags fix their requiredness, and the response codes fix the permissible values.

The value-level metadata rules are the heart of the Define-XML document. Specialization variables with a comparator of EQ or IN become the conditions of a WhereClause; specialization variables with vlmTarget = true become the variables described under that WhereClause, carrying their own data type, length, origin, and codelist/subset references. For example, in a laboratory (LB) group, a composite WhereClause such as (LBTESTCD EQ "BILI") AND (LBCAT EQ "CHEMISTRY") AND (LBSPEC EQ "BLOOD") describes the LBORRES/LBORRESU value-level metadata for that test; in a vital signs (VS) group, VSTESTCD EQ "SYSBP" describes the VSORRES/VSORRESU metadata for systolic blood pressure. Codelists referenced by the retained value-level metadata are collected into the codelist metadata. Their members come from 2 complementary sources: the specialization's value lists and subsetCodelist references retrieved from the CDISC Library, and the enabled response codes recorded on the BC properties in the USDM. Each enabled response code contributes 1 codelist item—coded value, decode, and NCI C-code—and when the enabled set is narrower than the full CT codelist, the

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

result is a subset codelist that presents only the study's permissible values. NCI C-codes used in value-level metadata are retrieved from the CDISC Library as they are not included in the SDTM DSS.

This value-level metadata logic is exactly the mechanism Jansen[1] described for building Define-XML v2.1 from SDTM DSS. The USDM adds the step that this earlier paper described being performed manually: identifying which specializations are in scope, since the study's activities reference the BCs those specializations implement.

Note: SDTM trial design and subject-level domains (TS, TA, TE, TI, TV, SV, SE, IE) are seeded directly from USDM study-design content and not from a union of specializations. USDM-HB1 describes creating the trial design domains, including the corresponding Define-XML input.

Step 5: Metadata Assembly – Assembling Define-XML Metadata

The assembled metadata are organized as a set of logical metadata structures that correspond directly to the major element groups of a Define-XML document: study, standards, dataset, variable, value-level, codelist, and document metadata. This organization is implementation-neutral—structures can be held in a relational database, in JSON or another serialization, in an MDR, or in the working tables of a particular toolkit—and is described here independently of any product. Each structure is populated from a defined source:

Define-XML metadata structure	Content	Primary source
Study metadata	Study name, description, protocol name	USDM study version and identifiers
Standards metadata	SDTMIG/CDASHIG and CT versions used	Versions are set by an external parameter setting used by the software
Dataset (ItemGroup) metadata	Dataset name, domain, label, class, structure, keys, purpose	SDTM DSS and CDISC Library SDTMIG (scoped by the DSS domains)
Variable (Item) metadata	Variable label, order, datatype, length, codelist, origin, format, derivation	SDTM DSS and CDISC Library SDTMIG
Value-level metadata	WhereClause condition plus the described variable's attributes	SDTM DSS (comparator, vlmTarget)
Codelist metadata	Codelists and subset codelists: terms, decodes, NCI codes, external terminologies	CDISC CT + specialization valueList/subset
Document metadata	Referenced documents (annotated CRF, reviewer guides) and links	Sponsor-supplied operational metadata

Note: These 7 structures map one-to-one onto the source metadata datasets used in the Jansen[1] reference implementation (source_study, source_standards, source_tables, source_columns, source_values, source_codelists, source_documents); that naming is one concrete realization of the implementation-neutral model described here.

Step 6: XML Generation and Rendition

After completing the initial Define-XML document generation, a refinement process adds any missing study-specific metadata. Once the metadata is complete, the Define-XML document is serialized from it, and an HTML rendition is typically produced via a Define-XML stylesheet, most commonly the stylesheet published with the standard. This is a well-defined, language-independent transformation: Any environment that can read the assembled metadata and write XML (e.g., Python, Java, R, SAS) can perform it. Generation is deterministic: The same assembled metadata yields the same document. The transformation is followed by schema validation and conformance rule checking; the specific toolkit used is an implementation detail and is not prescribed by this handbook.

4.2 Element-level Mapping

The following table summarizes how the principal USDM and specialization elements map to Define-XML constructs. It is illustrative rather than exhaustive.

Source	Source content	Define-XML target
USDM	StudyVersion identifiers, titles; study name/description	Study-level metadata (Study / MetaDataVersion attributes)
External, such as a command-line argument	Standards metadata (SDTMIG, CT versions)	Standards element and def:Standards

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Source	Source content	Define-XML target
SDTM DSS	Union of specialization domains	ItemGroupDef (one per SDTM dataset/domain)
SDTMIG CDISC Library API or Library exports	SDTMIG variable list for each in-scope domain	ItemDef / ItemRef within each ItemGroupDef
SDTM DSS and external	Specialization variable dataType/length/format/significantDigits	ItemDef @DataType, @Length, def:DisplayFormat, @SignificantDigits
SDTM DSS and external	Specialization originType / originSource	def:Origin (@Type, source)
SDTM DSS	Specialization comparator (EQ/IN) variables	def:WhereClauseDef conditions (RangeCheck)
SDTM DSS	Specialization vlmTarget variables	def:ValueListDef / ItemDef value-level definitions
SDTM DSS and CT; USDM first, then CDISC Library API	Codelist / subsetCodelist / valueList; CT terms	CodeList and CodeListItem (with subset codelists), RangeCheck/CheckValue
External	Annotated CRF and reviewer guide references (sponsor supplied)	def:leaf / def:DocumentRef

4.3 Supplying Study-Specific Metadata Not in the Standard Sources

Even with the USDM, SDTM DSS, and CDISC CT in hand, the Define-XML is not yet complete. A residual set of metadata is neither part of the study's conceptual definition nor part of any published standard, because it describes how a particular study was conducted and documented. It must be supplied from elsewhere, and deciding where each element comes from—and which source prevails when more than one could provide it—is the last design decision in the pipeline.

The residual metadata typically includes study-specific metadata, such as missing

- Variable lengths
- Significant digits
- Display formats
- Origin source, type, and links to specific annotated CRF pages
- Key sequences
- Derivation or computation method descriptions for derived and assigned variables
- Comments
- External dictionary and its version recorded in an ExternalCodeList (e.g., MedDRA or WHODrug release used to code events and medications)
- Study-specific codelist subsets
- Physical dataset references that include the leaf href to each dataset file
- Study-specific datasets not provided via the USDM study design
- Study-specific variables added to existing datasets

There is no single right way to supply this metadata; the following options can be combined, and each involves different trade-offs among effort, reuse, and governance.

- **A study-specific metadata plugin.** The generator exposes a defined extension point through which a study-specific metadata module is supplied and merged into the assembled metadata before serialization. The plugin is a self-contained artifact—a file or a service conforming to an agreed-upon schema—that provides exactly the residual elements (lengths, methods, comments, dictionary versions, document references) keyed to the datasets, variables, and value-level entries to which they apply. This keeps the standard-derived content untouched and machine generated, isolates the hand-authored study content in a single governed place, and lets a single generator serve every study by swapping plugins. Its cost is that the plugin needs a clear contract—which keys it supplies and how it takes precedence—and its own review,

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

since it carries the metadata a reviewer is most likely to scrutinize. The 360i Data Definition Engine project uses such a mechanism and can iteratively apply the plugin until completion.

- **A sponsor MDR for company defaults.** Much of the residual metadata is not truly per study but per sponsor: standard comment and method libraries, default lengths and length conventions, external dictionary versions currently in use, and annotated-CRF conventions. Holding these in an MDR and drawing on them at generation time reduces the per-study effort to genuinely unique cases and enforces consistency across a program. The repository and the plugin are complementary; the repository supplies defaults and the plugin overrides them where a study differs.
- **Derivation from produced artifacts.** Some residual metadata need not be authored at all. For example, where the annotated CRF is itself generated or digitally mapped, variable-to-page references can be derived from that mapping rather than transcribed. Additionally, CRFs and DTAs often include variables (e.g., VSSTAT/VSREASND) that may be added to the SDTM datasets but which are not available in the BCs and SDTM DSS.
- **USDM extension attributes.** The USDM may already carry the specialization link as an extension attribute, and the same mechanism can carry selected residual metadata so that it travels with the study definition rather than in a separate file. For example, adding the origin source may also benefit other downstream processes such as EDC setup and DTA specifications. Additionally, method descriptions tied to activities or document references derived from the CRF design are natural candidates. This preserves a single source of truth, but it is limited to metadata that genuinely belongs to the study definition and depends on the authoring solution supporting the extensions.

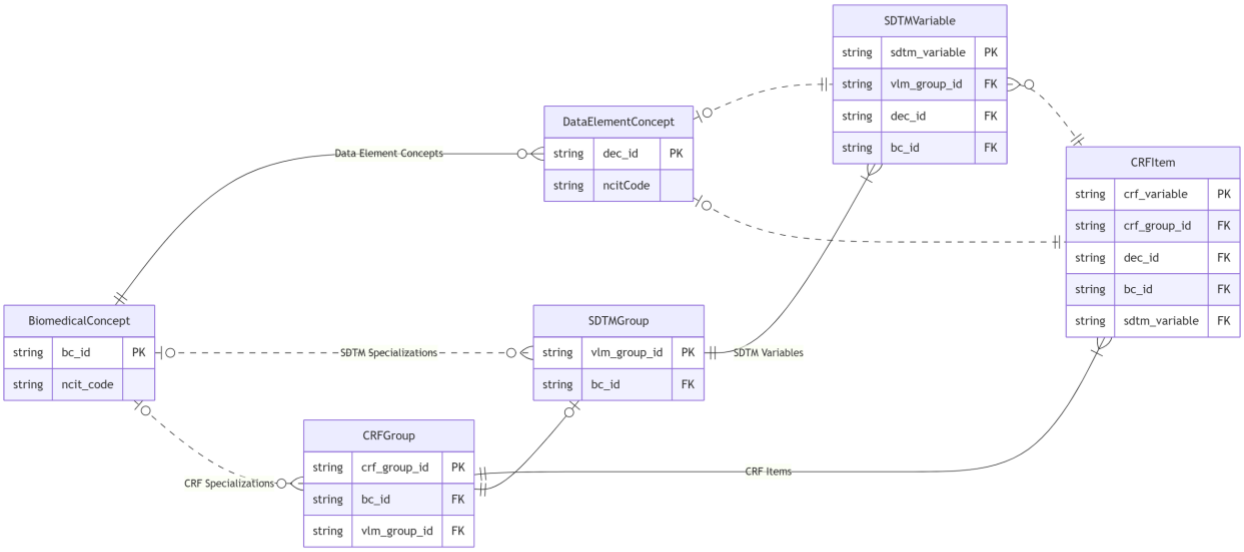
In practice, these options are layered rather than exclusive. A workable arrangement is to derive what can be derived from the data and the CRF, hold company-standard defaults in an MDR, and use a study-specific plugin for the remaining unique values—treating the merged result as the operational specification (the DDS layer of the reference architecture). Whichever combination is chosen, 2 rules matter: every residual element should have a single defined owner and source of truth, and the generator should apply an explicit precedence order when more than 1 source offers the same attribute (e.g., the study plugin overrides the repository default, which in turn overrides the specialization's nominal value). Making that precedence explicit is what keeps regeneration deterministic and auditable. The following table summarizes typical residual elements and where each is best supplied.

Residual metadata element	Why the standard sources omit it	Recommended source
Actual variable length, significant digits, and display format	USDM/specialization provides nominal length, significant digits, and display format values	The study plugin provides missing lengths or lengths that require study-specific values, as well as significant digits and display format values.
Method/derivation descriptions (def:MethodDef)	Specific to the study's derivations	Study plugin, drawing on a repository library of standard methods
Comments (def:CommentDef)	Not part of concept or standard metadata	Repository defaults, overridden by the study plugin
External dictionary + version (ExternalCodeList)	CDISC CT covers CDISC codelists, not MedDRA/WHODrug	Study/config value, ideally from a repository of dictionary versions; else study plugin
Annotated CRF page references (def:leaf, PDFPageRef)	Depends on the study's aCRF document	Derive from the CRF design where available; else study plugin
Dataset file references (leaf href), keys, sort order	Physical/operational, not conceptual	Generation-time configuration; else study plugin
Origin source and type values that are not provided by the DSS	Not all variables have static origin types and sources, so they cannot be set in the DSS.	Missing origin types and sources are provided by the study plugin, or may be generated dynamically based on the aCRF and other study metadata.
Additional variables in datasets (ItemDef/ItemRef)	Not every variable that should be added to SDTM datasets is included in the BC/DSSs, such as VSSTAT/VSREASND.	CRFs and DTAs are good sources for additional variables collected in the study that may not be represented in BCs and SDTM DSS but should still be included in the SDTM datasets. The SDTMIG may include expected or required variables for a given dataset that are not included in the SDTM DSS.
Additional datasets (ItemGroupDef)	Not every SDTMIG dataset needed for a study may be referenced in the USDM metadata.	The SDTMIG references datasets that are not included in the USDM, such as SPECIAL

The BCs in the USDM provide access to the SDTM DSS that determine which SDTM domains and variables appear in the Define-XML, but these same USDM BCs may also link to CRF specializations that provide data collection

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

details. The data collection details may also provide study-specific metadata that can be used to enrich the Define-XML. The following figure shows how BCs relate to both SDTM DSS and CRF specializations. The data element concepts are stored in USDM as properties and are directly related to the corresponding SDTM variable and CRF collection item.



4.4 Validation

A generated Define-XML must be validated at 2 levels before use, and its consistency with the datasets it describes must be confirmed at defined quality gates.

Validation Checks

Validation proceeds from the syntactic to the semantic. The first check is schema compliance: The document must validate against the Define-XML schema, which confirms syntactic correctness but cannot confirm full conformance to the specification. The full conformance check requires evaluating the define.xml against the published Define-XML conformance rules, which cover many consistency and completeness requirements that the schema alone does not enforce. Beyond the document in isolation, the described dataset and variable metadata must be consistent with the applicable SDTMIG and, where the datasets themselves exist, with those datasets. A Define-XML that describes variables the data does not contain, or omits variables it does, is not fit for submission. Finally, validate controlled terminology: The codelists and subset codelists must reference the correct CT package and contain valid terms and NCI codes, since a generated document inherits the correctness of the terminology it was built from. Tools that generate a Define-XML specification may also include these validation steps to ensure the final deliverable is valid and conformant to the standard.

Quality Gates

These checks apply at defined points in the study lifecycle rather than only at the end. A presubmission validation gate applies schema and conformance-rule validation before the Define-XML is included in any submission package, so that no nonconformant document leaves the organization. At database lock, a cross-check reconciles the Define-XML metadata against the SDTM dataset metadata (variable presence, data types, lengths, and codelists) to confirm that the description and the data have not diverged. Before any regulated use, a human-in-the-loop review gate requires a reviewer to confirm the generated metadata, paying particular attention to operational metadata the

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

USDM does not supply, where generation gives way to human judgment. Together, these gates ensure that automation accelerates the work without removing the accountable checkpoints a regulated process depends on.

System Architecture and Validation

The System Context and Architecture figure shows where the validation steps fit within the architecture layers. Before generating the define.xml, software applies USDM conformance checks to ensure the model conforms to the standard. Define.xml validation checks initially occur in layer 4, where software tools apply schema validation and conformance rules to confirm the state of the initial Define-XML document. Layer 4 also begins the human-in-the-loop reviews to make any necessary refinements to the define.xml. Layer 5 highlights the submission validation step that evaluates the define.xml generated in layer 4 against the actual study and addresses cases where divergence has occurred. The submission validation step also re-applies those checks performed in layer 4.

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

5 Challenges and Considerations

USDM-driven Define-XML generation is powerful, but requires additional sources of study-specific metadata to generate a complete Define-XML. That said, this approach provides the metadata needed to generate the bulk of the Define-XML, including the most challenging parts (e.g., value-level metadata). It also provides the means to generate the Define-XML based on the study design in the early stages of the study.

The challenges of implementing a USDM-driven Define-XML solution fall into 3 groups: the additional study-specific metadata required to complete the USDM-driven Define-XML generation process, the organizational shifts this approach requires, and the strategic tensions that arise from adopting a standard that is still maturing. None of these is a reason to defer adoption, but each should be planned for.

Technical

Study-specific Metadata

The most fundamental technical challenge is that the USDM requires additional study-specific metadata to generate a complete Define-XML. The required additional metadata has been documented in preceding sections of this handbook, along with strategies for providing it. This additional metadata must be properly governed to ensure it remains accurate across regenerations.

Process and Metadata Variability

A related difficulty is the variability in how sponsors implement SDTM: dataset splitting, custom domains, and local conventions for supplemental qualifiers all complicate a fully generic mapping, so any implementation must allow for sponsor-specific extension without abandoning the standard content it generates. Although such variability and sponsor-specific extensions make the Define-XML generation process more complex, the ideas proposed in this handbook still apply.

Concepts Surrogates and Dataset Specializations

An activity may reference a BC or concept surrogate for which no published SDTM DSS exists; for these assessments, no value-level metadata can be generated automatically, and the corresponding DSS metadata must be authored by the implementer (or a specialization proposed to CDISC). Surrogates make these gaps explicit, but without the dataset specialization, they do not close them.

Specializations and Standard Versions

The versions in play for a DSS must be reconciled: the SDTMIG range declared by a specialization, the CT package identified by the study's standards metadata, and the SDTMIG version the sponsor is targeting will not always coincide, and the generation process must detect and resolve such mismatches rather than silently producing an inconsistent document.

These versions must also be supported by the receiving regulator for the planned submission. Both the FDA and PMDA maintain data standards catalogs that identify the standards, terminologies, and versions they support. As a result, internal agreement is necessary, but it is not sufficient on its own.

Organizational

The larger organizational challenge is cultural rather than technical. The approach only pays off when study metadata is treated as a by-product of study design rather than as a deliverable assembled at the end, and that requires functions which have traditionally worked in sequence—design, data management, programming, standards—to work from a shared, upstream source. This is a change in procedure as much as in tooling, and it typically requires revising the supporting standard operating procedures for protocol authoring, review, and amendment. Tool fragmentation compounds the difficulty: Authoring, standards management, and submission are often supported by different systems that do not exchange metadata cleanly, and bridging or consolidating them is part of the cost of adoption.

Strategic

There is an underlying tension that recurs throughout digital data flow: the pull between flexibility, which allows a sponsor to reflect its own practice, and standardization, which makes reuse and automation possible in the first

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

place. Leaning too far toward flexibility erodes the very consistency the approach is meant to deliver; leaning too far toward standardization can produce metadata that does not match how the sponsor actually runs studies. Managing that balance deliberately, rather than resolving it once and for all, is itself a strategic capability.

Achieving this balance requires ongoing organizational commitment, not a one-time design decision. The benefits of standardization (e.g., reusing BCs) only accrue when the structured study definition remains authoritative and is not eroded by study-specific deviations. Further, standards governance must do more than review Define-XML output. It must continually decide where standardization is required and where flexibility is warranted. A fit-for-purpose process that evaluates deviations against the organization's maturity, as described in Section 2.4, Maturity Matrix, provides a repeatable way to make those decisions and manage the tension between flexibility and standardization.

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

6 Examples and Use Cases

The process described in preceding sections is deliberately implementation neutral: It consists of metadata retrieval, transformation, and serialization steps that can be built on any modern language or platform (e.g., Python, Java, R, SAS) and which consume the same CDISC Library API and the same specialization and terminology content regardless of technology. Several components illustrate the process end to end and are referenced here as demonstrations rather than as production-grade or endorsed tools. For example, the CDISC 360i Data Definition Engine (DDE, Python; https://www.cdisc.org/standards/cdisc-360i) and the SAS Clinical Standards Toolkit (openCST; https://github.com/sassoftware/clinical-standards-toolkit) share the same underlying logic, which is language independent and reproduced in this handbook without reliance on any specific product. The 360i DDE project (https://github.com/cdisc-org/data-definition-engine), associated with the DDF initiative (https://transcelerate.github.io/ddf-home/index.html), demonstrates how to generate a complete Define-XML using a USDM-driven process. Any conformant implementation may be substituted: the USDM handbooks prescribe the metadata and the process, not the toolset.

This section illustrates the process with a real USDM study definition, NCT01797120 (protocol PrE0102), a randomized, double-blind, placebo-controlled phase 2 oncology trial of fulvestrant. The example is drawn from the USDM file exported by the open-source SOA Workbench (https://cosa.cdisc.org) and is used here to trace the metadata chain from study activities to Define-XML content.

The Study Definition

The study version contains a single study design with 21 activities. Its activities are bound to BCs: of 50 BCs in the study version, 43 carry an SDTM DSS reference; a further 14 concept surrogates represent assessments for which no published concept was available. Activities reference their concepts via biomedicalConceptIds; for example, the INFORMED CONSENT activity references the Informed Consent concept, PHYSICAL EXAM references Physical Examination Finding, and the laboratory and vital-signs activities reference the corresponding measurement concepts.

From Activity to Specialization

Each referenced BC resolves to an SDTM DSS, and the union of their domains determines the Define-XML dataset scope. The concepts in this study span, among others, the Disposition, Vital Signs, Laboratory Test Results, Concomitant/Prior Medications, Tumor/Lesion Identification, and Disease Response and Clin Classification datasets.

Activity/concept	Specialization	Domain
Informed consent	CONSENT	DS
Systolic blood pressure	SYSBP	VS
Body temperature	TEMP	VS
Total bilirubin measurement	BILISERPL	LB
Creatinine measurement	CREATSERPL	LB
Leukocyte count	WBCBLD	LB
Concomitant therapy	CMFREE	CM
Tumor identification	TUMERGE	TU
Overall response	OVRLRESP	RS

From Specialization to Value-level Metadata

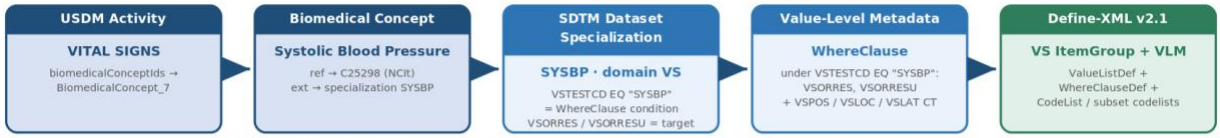
The following figure presents a worked trace from vital signs activity to VS value-level metadata, with the per-variable selection decision. The SYSBP, DIABP, HR, and TEMP specializations all target the VS domain. In each, the topic variable VSTESTCD carries a comparator (EQ) and an assigned term (e.g., SYSBP), so it becomes the WhereClause condition. Qualifier variables such as VSPOS, VSLOC, and VSLAT carry codelists and, where present, extend the condition into a composite WhereClause; VSORRES and VSORRESU carry vlmTarget = true, so they become the variables described by value-level metadata under that condition, each with its data type, length,

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

origin, and codelist. The result is a set of VS value-level metadata groups (1 per vital sign) assembled without transcription.

The same mechanism applied to the study's chemistry, hematology, and urinalysis concepts yields LB value-level metadata, with composite WhereClauses over LBTESTCD, LBCAT, and LBSPEC, and target variables LBORRES/LBORRESU; the tumor and response concepts yield TU and RS value-level metadata.

Worked Trace — Systolic Blood Pressure (study NCT01797120)



Variable selection for the VS domain — SDTMIG core × Biomedical Concept property flag

VS variable	SDTMIG core	Flag / status	In Define-XML?
VSTESTCD	Required	WhereClause condition — EQ "SYSBP"	IN
VSORRES	Expected	value-level metadata target	IN
VSORRESU	Expected	enabled (units)	IN
VSPOS / VSLOC / VSLAT	Permissible	isEnabled = true → selected	IN
VSDTC	Expected	enabled (date/time)	IN
VSGRPID, VSSPID, VSSTAT, …	Permissible	no property present → dropped	OUT

Required and Expected variables are included by the standard; Permissible variables are included only when a Biomedical Concept property enables them.

Controlled Terminology and Codelists

The codelists referenced by the retained value-level metadata (e.g., VS position, location, and laterality codelists; LB test and specimen codelists) are collected from the CDISC CT package identified by the study's standards metadata. Subset codelists are built from the specialization value lists so that the Define-XML presents only the terms that apply to the study. The following figure shows a Define-XML codelist subset for Vital Signs Position.

```
<CodeList OID="CL.POSITION" Name="Position" DataType="text" def:StandardOID="STD.SDTMCT">
    <CodeListItem CodedValue="SITTING">
        <Decode>
            <TranslatedText xml:lang="en">Sitting</TranslatedText>
        </Decode>
        <Alias Context="nci:ExtCodeID" Name="C62122"/>
    </CodeListItem>
    <CodeListItem CodedValue="STANDING">
        <Decode>
            <TranslatedText xml:lang="en">Orthostatic</TranslatedText>
        </Decode>
        <Alias Context="nci:ExtCodeID" Name="C62166"/>
    </CodeListItem>
    <CodeListItem CodedValue="SUPINE">
        <Decode>
            <TranslatedText xml:lang="en">Supine</TranslatedText>
        </Decode>
        <Alias Context="nci:ExtCodeID" Name="C62167"/>
    </CodeListItem>
    <Alias Context="nci:ExtCodeID" Name="C71148"/>
</CodeList>
```

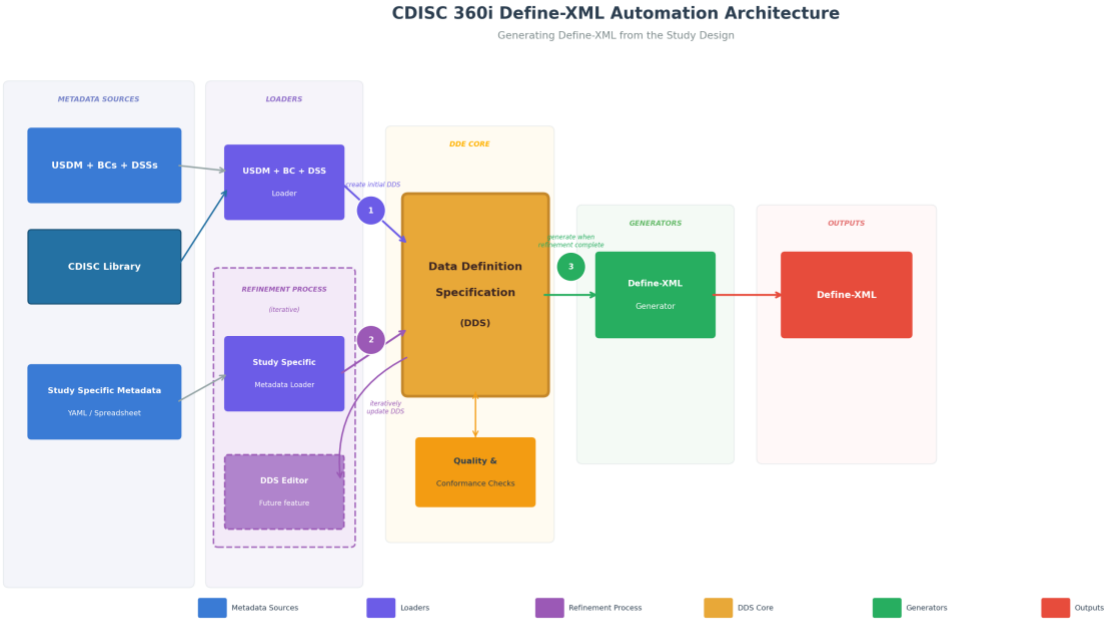
© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Complex-study Considerations

NCT01797120 is multi-arm and multivisit, and its oncology response assessments (target lesion, non-target lesion, overall response) exercise the full chain: activity grouping in the USDM, multiple concepts per activity, composite WhereClauses, and cross-domain terminology. It therefore demonstrates the process at a realistic scale—from a simple single-condition vital sign to a multicondition laboratory panel to grouped response assessments—while still resolving cleanly to Define-XML value-level metadata for concepts that have published specializations.

CDISC 360i Example

The Define-XML generation workstream in the CDISC 360i program created the DDE, which implements this use case (as well as others, such as CRF generation). The DDE drives the generation of a Define-XML from the USDM for NCT01797120. It uses the BCs referenced in the USDM SoA and looks up the associated SDTM DSS, as well as the SDTMIG and SDTM CT, to generate most of the Define-XML, including value-level metadata. The remaining study-specific metadata, previously labeled as a plugin, is added during the refinement process, as shown in the following figure.



This figure highlights the use of the DDS model as a repository for operational study metadata used to automate research processes. The Define-XML generation is one part of an overall automated study pipeline; other parts, such as aCRF generation and raw-to-SDTM transformations, are also supported by the DDS model.

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Appendices

Appendix A: USDM Example JSON

Appendix A1: Instantiated BC: Informed Consent

This example shows a study-instantiated BC for the informed consent collection. The relevant data element concepts from the conceptual BC library (BC C16735) are included as properties (see USDM-HB2 for mapping):

- Reported Event Term
- Dictionary-derived Term
- Category (disposition category)
- Observation Start Date Time

This directly reflects the corresponding elements in the SDTM DSS. The corresponding response codes are derived from the DSS and inform standard values in SDTM if only 1 value is specified as in Disposition (=PROTOCOL MILESTONE), and the dictionary-derived term (=INFORMED CONSENT OBTAINED).


```
"biomedicalConcepts": [
          {
            "id": "BiomedicalConcept_2",
            "extensionAttributes": [
                {
                    "id": "ExtensionAttributeValue_1",
                    "url": "//ExtensionLibrary/USDMv4_0/SDTMSpecialization/v1/reference",
                    "valueString": "/mdr/specializations/sdtm/packages/2024-12-
16/datasetspecializations/CONSENT",
                    "extensionAttributes": [],
                    "instanceType": "ExtensionAttribute"
                }],
            "name": "ICF",
            "label": "Informed Consent",
            "synonyms": [
              "Informed Consent Obtained"
            ],
            "reference": "//sponsor_mdr//ConfiguredBCs/v1/ICF1",
            "properties": [
              {
                "id": "BiomedicalConceptProperty_201",
                "extensionAttributes": [],
                "name": "ICF_DSCAT",
                "label": "ICF Milestone category",
                "isRequired": true,
                "isEnabled": true,
                "datatype": "text",
                "responseCodes": [{
                    "id": "ResponseCode_5_2",
                    "extensionAttributes": [],
                    "name": "DSCAT_ICF",
                    "label": "",
                    "isEnabled": true,
                    "code": {
                      "id": "Code_1561",
                      "extensionAttributes": [],
                      "code": "C74588",
                      "codeSystem": "http://www.cdisc.org",
                      "codeSystemVersion": "2025-09-26",
                      "decode": "PROTOCOL MILESTONE",
                      "instanceType": "Code"
                    },
                    "instanceType": "ResponseCode"
```


© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

```
        }],
      "code": {
        "id": "AliasCode_14",
        "extensionAttributes": [],
        "standardCode": {
          "id": "Code_143_2",
          "extensionAttributes": [],
          "code": "C25372",
          "codeSystem": "http://www.cdisc.org",
          "codeSystemVersion": "2025-09-26",
          "decode": "Dictionary-derived Term",
          "instanceType": "Code"
        },
        "standardCodeAliases": [],
        "instanceType": "AliasCode"
      },
      "notes": [],
      "instanceType": "BiomedicalConceptProperty"
    },
    {
      "id": "BiomedicalConceptProperty_5",
      "extensionAttributes": [],
      "name": "ICF_OBTAINED",
      "label": "Informed Consent Obtained",
      "isRequired": true,
      "isEnabled": true,
      "datatype": "text",
      "responseCodes": [{
          "id": "ResponseCode_5_1",
          "extensionAttributes": [],
          "name": "DSDECOD_ICF",
          "label": "",
          "isEnabled": true,
          "code": {
            "id": "Code_1561",
            "extensionAttributes": [],
            "code": "C16735",
            "codeSystem": "http://www.cdisc.org",
            "codeSystemVersion": "2025-09-26",
            "decode": "INFORMED CONSENT OBTAINED",
            "instanceType": "Code"
          },
          "instanceType": "ResponseCode"
        }],
      "code": {
        "id": "AliasCode_14",
        "extensionAttributes": [],
        "standardCode": {
          "id": "Code_143",
          "extensionAttributes": [],
          "code": "C82977",
          "codeSystem": "http://www.cdisc.org",
          "codeSystemVersion": "2025-09-26",
          "decode": "Dictionary-derived Term",
          "instanceType": "Code"
        },
        "standardCodeAliases": [],
        "instanceType": "AliasCode"
      },
      "notes": [],
      "instanceType": "BiomedicalConceptProperty"
    },
    {
      "id": "BiomedicalConceptProperty_6",
      "extensionAttributes": [],
      "name": "ICF_TERM",
      "label": "Reported Event Term",
      "isRequired": true,
```


© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

```
            "isEnabled": true,
            "datatype": "text",
            "responseCodes": [],
            "code": {
              "id": "AliasCode_15",
              "extensionAttributes": [],
              "standardCode": {
                "id": "Code_144",
                "extensionAttributes": [],
                "code": "C82571",
                "codeSystem": "http://www.cdisc.org",
                "codeSystemVersion": "2025-09-26",
                "decode": "Reported Event Term",
                "instanceType": "Code"
              },
              "standardCodeAliases": [],
              "instanceType": "AliasCode"
            },
            "notes": [],
            "instanceType": "BiomedicalConceptProperty"
          },
          {
            "id": "BiomedicalConceptProperty_7",
            "extensionAttributes": [],
            "name": "ICF_STDTC",
            "label": "ICF Date",
            "isRequired": true,
            "isEnabled": true,
            "datatype": "datetime",
            "responseCodes": [],
            "code": {
              "id": "AliasCode_16",
              "extensionAttributes": [],
              "standardCode": {
                "id": "Code_145",
                "extensionAttributes": [],
                "code": "C82517",
                "codeSystem": "http://www.cdisc.org",
                "codeSystemVersion": "2025-09-26",
                "decode": "Observation Start Date Time",
                "instanceType": "Code"
              },
              "standardCodeAliases": [],
              "instanceType": "AliasCode"
            },
            "notes": [],
            "instanceType": "BiomedicalConceptProperty"
          }
        ],
        "code": {
          "id": "AliasCode_17",
          "extensionAttributes": [],
          "standardCode": {
            "id": "Code_146",
            "extensionAttributes": [],
            "code": "C16735",
            "codeSystem": "http://www.cdisc.org",
            "codeSystemVersion": "2025-09-26",
            "decode": "Informed Consent",
            "instanceType": "Code"
          },
          "standardCodeAliases": [],
          "instanceType": "AliasCode"
        },
        "notes": [],
        "instanceType": "BiomedicalConcept"
      } ]
```


© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Appendix A2: Instantiated BC: Platelet Count

This example shows a study-instantiated BC for the platelet count in blood collection. The relevant data element concepts from the conceptual BC library (BC C51951) are included as properties (see USDM-HB2 for mapping):

- Observation Result
- Unit of number Concentration
- Collection Date Time
- Fasting Status Indicator
- Biospecimen Type

This directly reflects the corresponding elements in the SDTM DSS. More elements are allowed to be added in USDM to inform downstream processes. In order to add the correct LBCAT, LBTESTCD, and LBTEST variables, these were added as properties as well, with standard values stored as response codes. This ensures alignment with the DSS and/or directly informs the SDTM Define-XML that the codes for platelets in blood need to be added (LBCAT=HEMATOLOGY, LBTEST=Platelets, LBTESTCD=PLAT).

A standard unit is specified without other options (as chosen for this instantiation), and a standard specimen was configured to align with the blood collection. The fasting status has still 2 options which are enabled (Yes and No); this informs downstream systems that multiple options should be enabled and in SDTM Define-XML all options need to be included. However, if the fasting status is not relevant/applicable to the assessment, then the whole property can be removed or isRequired and isEnabled can be set to false.

Note that the corresponding LOINC code is specified as an alias code for the CDISC C-code. This allows for direct downstream connection and mapping. More alternative codes can be added as needed, either from the same code system (when more LOINC codes are applicable to the instantiated BC) or other code systems that might be applicable for the data collection.


```
"biomedicalConcepts": [{
            "id": "BiomedicalConcept_13",
            "extensionAttributes": [],
            "name": "PLATBLD1",
            "label": "Platelet Count in Blood",
            "synonyms": [
              "Thrombocytes",
              "Platelets",
              "Anucleated Thrombocytes",
              "Platelet Count"
            ],
            "reference": "//sponsor_mdr//ConfiguredBCs/v1/PLATBLD1",
            "properties": [
              {
                "id": "BiomedicalConceptProperty_69a",
                "extensionAttributes": [],
                "name": "LBTESTCD_PLATBLD",
                "label": "LBTESTCD",
                "isRequired": true,
                "isEnabled": true,
                "datatype": "text",
                "responseCodes": [{
                    "id": "ResponseCode_5_5",
                    "extensionAttributes": [],
                    "name": "LBTESTCD_PLATBLD",
                    "label": "",
                    "isEnabled": true,
                    "code": {
                      "id": "Code_1563",
                      "extensionAttributes": [],
                      "code": "C51951",
                      "codeSystem": "http://www.cdisc.org",
                      "codeSystemVersion": "2025-09-26",
```


© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

```
            "decode": "PLAT",
            "instanceType": "Code"
          },
          "instanceType": "ResponseCode"
        }],
      "code": {
        "id": "AliasCode_89",
        "extensionAttributes": [],
        "standardCode": {
          "id": "Code_279",
          "extensionAttributes": [],
          "code": "C65047",
          "codeSystem": "http://www.cdisc.org",
          "codeSystemVersion": "2025-09-26",
          "decode": "Laboratory Test Code Terminology",
          "instanceType": "Code"
        },
        "standardCodeAliases": [],
        "instanceType": "AliasCode"
      },
      "notes": [],
      "instanceType": "BiomedicalConceptProperty"
    },
    {
      "id": "BiomedicalConceptProperty_69b",
      "extensionAttributes": [],
      "name": "LBTEST_PLATBLD",
      "label": "LBTEST",
      "isRequired": true,
      "isEnabled": true,
      "datatype": "text",
      "responseCodes": [{
          "id": "ResponseCode_5_6",
          "extensionAttributes": [],
          "name": "LBTEST_PLATBLD",
          "label": "",
          "isEnabled": true,
          "code": {
            "id": "Code_1564",
            "extensionAttributes": [],
            "code": "C51951",
            "codeSystem": "http://www.cdisc.org",
            "codeSystemVersion": "2025-09-26",
            "decode": "Platelets",
            "instanceType": "Code"
          },
          "instanceType": "ResponseCode"
        }],
      "code": {
        "id": "AliasCode_239",
        "extensionAttributes": [],
        "standardCode": {
          "id": "Code_601",
          "extensionAttributes": [],
          "code": "C67154",
          "codeSystem": "http://www.cdisc.org",
          "codeSystemVersion": "2025-09-26",
          "decode": "Laboratory Test Name Terminology",
          "instanceType": "Code"
        },
        "standardCodeAliases": [],
        "instanceType": "AliasCode"
      },
      "notes": [],
      "instanceType": "BiomedicalConceptProperty"
    },
    {
      "id": "BiomedicalConceptProperty_70",
```


© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

```
      "extensionAttributes": [],
      "name": "LBORRES_PLATBLD",
      "label": "LBORRES",
      "isRequired": true,
      "isEnabled": true,
      "datatype": "float",
      "responseCodes": [],
      "code": {
        "id": "AliasCode_90",
        "extensionAttributes": [],
        "standardCode": {
          "id": "Code_280",
          "extensionAttributes": [],
          "code": "C70856",
          "codeSystem": "http://www.cdisc.org",
          "codeSystemVersion": "2025-09-26",
          "decode": "Observation Result",
          "instanceType": "Code"
        },
        "standardCodeAliases": [],
        "instanceType": "AliasCode"
      },
      "notes": [],
      "instanceType": "BiomedicalConceptProperty"
    },
    {
      "id": "BiomedicalConceptProperty_71",
      "extensionAttributes": [],
      "name": "LBORRESU_PLATBLD",
      "label": "LBORRESU",
      "isRequired": true,
      "isEnabled": true,
      "datatype": "text",
      "responseCodes": [
        {
          "id": "ResponseCode_65",
          "extensionAttributes": [],
          "name": "RC_C67255_65",
          "label": "",
          "isEnabled": true,
          "code": {
            "id": "Code_281",
            "extensionAttributes": [],
            "code": "C67255",
            "codeSystem": "http://www.cdisc.org",
            "codeSystemVersion": "2025-09-26",
            "decode": "Billion per Liter",
            "instanceType": "Code"
          },
          "instanceType": "ResponseCode"
        }
      ],
      "code": {
        "id": "AliasCode_91",
        "extensionAttributes": [],
        "standardCode": {
          "id": "Code_282",
          "extensionAttributes": [],
          "code": "C67391",
          "codeSystem": "http://www.cdisc.org",
          "codeSystemVersion": "2025-09-26",
          "decode": "Unit of Number Concentration",
          "instanceType": "Code"
        },
        "standardCodeAliases": [],
        "instanceType": "AliasCode"
      },
      "notes": [],
```


© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

```
      "instanceType": "BiomedicalConceptProperty"
    },
    {
      "id": "BiomedicalConceptProperty_72",
      "extensionAttributes": [],
      "name": "LBSPEC_PLATBLD",
      "label": "LBSPEC",
      "isRequired": true,
      "isEnabled": true,
      "datatype": "text",
      "responseCodes": [
        {
          "id": "ResponseCode_153",
          "extensionAttributes": [],
          "name": "RC_C12434_01",
          "label": "",
          "isEnabled": true,
          "code": {
            "id": "Code_576",
            "extensionAttributes": [],
            "code": "12434",
            "codeSystem": "http://www.cdisc.org",
            "codeSystemVersion": "2025-09-26",
            "decode": "BLOOD",
            "instanceType": "Code"
          },
          "instanceType": "ResponseCode"
        }],
      "code": {
        "id": "AliasCode_92",
        "extensionAttributes": [],
        "standardCode": {
          "id": "Code_283",
          "extensionAttributes": [],
          "code": "C70713",
          "codeSystem": "http://www.cdisc.org",
          "codeSystemVersion": "2025-09-26",
          "decode": "Biospecimen Type",
          "instanceType": "Code"
        },
        "standardCodeAliases": [],
        "instanceType": "AliasCode"
      },
      "notes": [],
      "instanceType": "BiomedicalConceptProperty"
    },
    {
      "id": "BiomedicalConceptProperty_73",
      "extensionAttributes": [],
      "name": "LBFAST_PLATBLD",
      "label": "LBFAST",
      "isRequired": true,
      "isEnabled": true,
      "datatype": "text",
      "responseCodes": [
        {
          "id": "ResponseCode_66",
          "extensionAttributes": [],
          "name": "RC_C49487_66",
          "label": "",
          "isEnabled": true,
          "code": {
            "id": "Code_284",
            "extensionAttributes": [],
            "code": "C49487",
            "codeSystem": "http://www.cdisc.org",
            "codeSystemVersion": "2025-09-26",
            "decode": "No",
```


© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

```
            "instanceType": "Code"
          },
          "instanceType": "ResponseCode"
        },
        {
          "id": "ResponseCode_67",
          "extensionAttributes": [],
          "name": "RC_C49488_67",
          "label": "",
          "isEnabled": true,
          "code": {
            "id": "Code_285",
            "extensionAttributes": [],
            "code": "C49488",
            "codeSystem": "http://www.cdisc.org",
            "codeSystemVersion": "2025-09-26",
            "decode": "Yes",
            "instanceType": "Code"
          },
          "instanceType": "ResponseCode"
        }
      ],
      "code": {
        "id": "AliasCode_93",
        "extensionAttributes": [],
        "standardCode": {
          "id": "Code_286",
          "extensionAttributes": [],
          "code": "C93566",
          "codeSystem": "http://www.cdisc.org",
          "codeSystemVersion": "2025-09-26",
          "decode": "Fasting Status Indicator",
          "instanceType": "Code"
        },
        "standardCodeAliases": [],
        "instanceType": "AliasCode"
      },
      "notes": [],
      "instanceType": "BiomedicalConceptProperty"
    },
    {
      "id": "BiomedicalConceptProperty_181",
      "extensionAttributes": [],
      "name": "LBDTC_PLATBLD",
      "label": "LBDTC",
      "isRequired": true,
      "isEnabled": true,
      "datatype": "datetime",
      "responseCodes": [],
      "code": {
        "id": "AliasCode_240",
        "extensionAttributes": [],
        "standardCode": {
          "id": "Code_603",
          "extensionAttributes": [],
          "code": "C82515",
          "codeSystem": "http://www.cdisc.org",
          "codeSystemVersion": "2025-09-26",
          "decode": "Collection Date Time",
          "instanceType": "Code"
        },
        "standardCodeAliases": [],
        "instanceType": "AliasCode"
      },
      "notes": [],
      "instanceType": "BiomedicalConceptProperty"
    },
    {
```


© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

```
      "id": "BiomedicalConceptProperty_182",
      "extensionAttributes": [],
      "name": "LBCAT_PLATBLD",
      "label": "LBCAT",
      "isRequired": true,
      "isEnabled": true,
      "datatype": "text",
      "responseCodes": [{
          "id": "ResponseCode_5_5",
          "extensionAttributes": [],
          "name": "LBCAT_PLATBLD",
          "label": "",
          "isEnabled": true,
          "code": {
            "id": "Code_1563",
            "extensionAttributes": [],
            "code": "C49286",
            "codeSystem": "http://www.cdisc.org",
            "codeSystemVersion": "2025-09-26",
            "decode": "HEMATOLOGY",
            "instanceType": "Code"
          },
          "instanceType": "ResponseCode"
        }],
      "code": {
        "id": "AliasCode_89",
        "extensionAttributes": [],
        "standardCode": {
          "id": "Code_604",
          "extensionAttributes": [],
          "code": "C83017",
          "codeSystem": "http://www.cdisc.org",
          "codeSystemVersion": "2025-09-26",
          "decode": "Laboratory Test Category",
          "instanceType": "Code"
        },
        "standardCodeAliases": [],
        "instanceType": "AliasCode"
      },
      "notes": [],
      "instanceType": "BiomedicalConceptProperty"
    }
  ],
  "code": {
    "id": "AliasCode_94",
    "extensionAttributes": [],
    "standardCode": {
      "id": "Code_287",
      "extensionAttributes": [],
      "code": "C51951",
      "codeSystem": "http://www.cdisc.org",
      "codeSystemVersion": "2025-09-26",
      "decode": "platelets",
      "instanceType": "Code"
    },
    "standardCodeAliases": [
        {
          "id": "Code_602",
          "extensionAttributes": [],
          "code": "26515-7",
          "codeSystem": "http://www.LOINC.org",
          "codeSystemVersion": "2.73",
          "decode": "platelets",
          "instanceType": "Code"
        }],
    "instanceType": "AliasCode"
  },
  "notes": [],
```


© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

```
            "instanceType": "BiomedicalConcept"
          }]
```


Appendix A3: Instantiated BC: Weight

This example shows a study-instantiated BC for weight collection. The relevant data element concepts from the conceptual BC library (BC C81328) are included as properties (see USDM-HB2 for mapping):

- Observation Result
- Unit of Weight
- Collection Date Time

This directly reflects the corresponding elements in the SDTM DSS. More elements are allowed to be added in USDM to inform downstream processes. To add the correct VSTESTCD and VSTEST variables, these were added as properties as well with standard values stored as response codes. This ensures alignment with the DSS and/or directly informs the SDTM Define-XML that the codes for weight need to be added (VSTESTCD=WEIGHT, VSTEST=Weight).

For the unit, in this case only the protocol-indicated standard unit is enabled as response code (VSORRESU="Kilogram" / isEnabled=true). Other units are still available with the corresponding codes and can be used in case (due to a protocol deviation) another unit was specified.


```
"biomedicalConcepts": [
          {
            "id": "BiomedicalConcept_4",
            "extensionAttributes": [],
            "name": "WGHT",
            "label": "Weight",
            "synonyms": [
              "WEIGHT"
            ],
            "reference": "//sponsor_mdr//ConfiguredBCs/v1/WGHT1",
            "properties": [
              {
                "id": "BiomedicalConceptProperty_12",
                "extensionAttributes": [],
                "name": "VSTESTCD",
                "label": "SDTM VSTESTCD variable",
                "isRequired": true,
                "isEnabled": true,
                "datatype": "text",
                "responseCodes": [{
                    "id": "ResponseCode_5_3",
                    "extensionAttributes": [],
                    "name": "TESTCD_WEIGHT",
                    "label": "",
                    "isEnabled": true,
                    "code": {
                      "id": "Code_1562",
                      "extensionAttributes": [],
                      "code": "C25208",
                      "codeSystem": "http://www.cdisc.org",
                      "codeSystemVersion": "2025-09-26",
                      "decode": "WEIGHT",
                      "instanceType": "Code"
                    },
                    "instanceType": "ResponseCode"
                  }],
                "code": {
                  "id": "AliasCode_23",
                  "extensionAttributes": [],
```


© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

```
      "standardCode": {
        "id": "Code_152",
        "extensionAttributes": [],
        "code": "C66741",
        "codeSystem": "http://www.cdisc.org",
        "codeSystemVersion": "2025-09-26",
        "decode": "VSTESTCD",
        "instanceType": "Code"
      },
      "standardCodeAliases": [],
      "instanceType": "AliasCode"
    },
    "notes": [],
    "instanceType": "BiomedicalConceptProperty"
  },
  {
    "id": "BiomedicalConceptProperty_202",
    "extensionAttributes": [],
    "name": "VSTEST",
    "label": "SDTM VSTEST variable",
    "isRequired": true,
    "isEnabled": true,
    "datatype": "text",
    "responseCodes": [{
        "id": "ResponseCode_5_4",
        "extensionAttributes": [],
        "name": "TEST_WEIGHT",
        "label": "",
        "isEnabled": true,
        "code": {
          "id": "Code_1562",
          "extensionAttributes": [],
          "code": "C25208",
          "codeSystem": "http://www.cdisc.org",
          "codeSystemVersion": "2025-09-26",
          "decode": "Weight",
          "instanceType": "Code"
        },
        "instanceType": "ResponseCode"
      }],
    "code": {
      "id": "AliasCode_2301",
      "extensionAttributes": [],
      "standardCode": {
        "id": "Code_1521",
        "extensionAttributes": [],
        "code": "C67153",
        "codeSystem": "http://www.cdisc.org",
        "codeSystemVersion": "2025-09-26",
        "decode": "VSTEST",
        "instanceType": "Code"
      },
      "standardCodeAliases": [],
      "instanceType": "AliasCode"
    },
    "notes": [],
    "instanceType": "BiomedicalConceptProperty"
  },
  {
    "id": "BiomedicalConceptProperty_13",
    "extensionAttributes": [],
    "name": "VSORRES",
    "label": "Observation Result",
    "isRequired": true,
    "isEnabled": true,
    "datatype": "decimal",
    "responseCodes": [],
    "code": {
```


© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

```
        "id": "AliasCode_24",
        "extensionAttributes": [],
        "standardCode": {
          "id": "Code_153",
          "extensionAttributes": [],
          "code": "C70856",
          "codeSystem": "http://www.cdisc.org",
          "codeSystemVersion": "2025-09-26",
          "decode": "Observation Result",
          "instanceType": "Code"
        },
        "standardCodeAliases": [],
        "instanceType": "AliasCode"
      },
      "notes": [],
      "instanceType": "BiomedicalConceptProperty"
    },
    {
      "id": "BiomedicalConceptProperty_14",
      "extensionAttributes": [],
      "name": "VSORRESU",
      "label": "Unit of Weight",
      "isRequired": true,
      "isEnabled": true,
      "datatype": "text",
      "responseCodes": [
        {
          "id": "ResponseCode_4",
          "extensionAttributes": [],
          "name": "RC_C48531_4",
          "label": "",
          "isEnabled": false,
          "code": {
            "id": "Code_154",
            "extensionAttributes": [],
            "code": "C48531",
            "codeSystem": "http://www.cdisc.org",
            "codeSystemVersion": "2025-09-26",
            "decode": "Pound",
            "instanceType": "Code"
          },
          "instanceType": "ResponseCode"
        },
        {
          "id": "ResponseCode_5",
          "extensionAttributes": [],
          "name": "RC_C48155_5",
          "label": "",
          "isEnabled": false,
          "code": {
            "id": "Code_155",
            "extensionAttributes": [],
            "code": "C48155",
            "codeSystem": "http://www.cdisc.org",
            "codeSystemVersion": "2025-09-26",
            "decode": "Gram",
            "instanceType": "Code"
          },
          "instanceType": "ResponseCode"
        },
        {
          "id": "ResponseCode_6",
          "extensionAttributes": [],
          "name": "RC_C28252_6",
          "label": "",
          "isEnabled": true,
          "code": {
            "id": "Code_156",
```


© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

```
                    "extensionAttributes": [],
                    "code": "C28252",
                    "codeSystem": "http://www.cdisc.org",
                    "codeSystemVersion": "2025-09-26",
                    "decode": "Kilogram",
                    "instanceType": "Code"
                  },
                  "instanceType": "ResponseCode"
                }
              ],
              "code": {
                "id": "AliasCode_25",
                "extensionAttributes": [],
                "standardCode": {
                  "id": "Code_157",
                  "extensionAttributes": [],
                  "code": "C48208",
                  "codeSystem": "http://www.cdisc.org",
                  "codeSystemVersion": "2025-09-26",
                  "decode": "Unit of Weight",
                  "instanceType": "Code"
                },
                "standardCodeAliases": [],
                "instanceType": "AliasCode"
              },
              "notes": [],
              "instanceType": "BiomedicalConceptProperty"
            },
            {
              "id": "BiomedicalConceptProperty_15",
              "extensionAttributes": [],
              "name": "VSDTC",
              "label": "VS Date",
              "isRequired": true,
              "isEnabled": true,
              "datatype": "datetime",
              "responseCodes": [],
              "code": {
                "id": "AliasCode_26",
                "extensionAttributes": [],
                "standardCode": {
                  "id": "Code_158",
                  "extensionAttributes": [],
                  "code": "C82515",
                  "codeSystem": "http://www.cdisc.org",
                  "codeSystemVersion": "2025-09-26",
                  "decode": "Collection Date Time",
                  "instanceType": "Code"
                },
                "standardCodeAliases": [],
                "instanceType": "AliasCode"
              },
              "notes": [],
              "instanceType": "BiomedicalConceptProperty"
            }
          ],
          "code": {
            "id": "AliasCode_27",
            "extensionAttributes": [],
            "standardCode": {
              "id": "Code_159",
              "extensionAttributes": [],
              "code": "C81328",
              "codeSystem": "http://www.cdisc.org",
              "codeSystemVersion": "2025-09-26",
              "decode": "Body Weight",
              "instanceType": "Code"
            },
```


© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

```
            "standardCodeAliases": [],
            "instanceType": "AliasCode"
          },
          "notes": [],
          "instanceType": "BiomedicalConcept"
        } ]
```


© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Appendix B: Implementation Handbook Team

Name	Institution/Organization
John Owen	Project Manager, CDISC
Anthony Chow	Technical Writer, CDISC
Berber Snoeijer	USDM Expert and Technical Writer, CDISC
Sam Hume	Use Case SME, CDISC
Lex Jansen	Use Case SME, CDISC
Lorraine Sobson	Copyeditor, CDISC

CDISC also would like to thank Jozef Aerts, Sherry Chou, and Marcelina Hungriafor their review of this implementation handbook.

The USDM-HB3 has been developed in partnership with TransCelerate Biopharma.

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Appendix C: Glossary and Abbreviations

The following abbreviations and terms are used in this document. Additional definitions can be found in the sections of this Implementation Handbook, in the USDM-IG v4.0, USDM-HB1 and USDM-HB2, and in the CDISC Glossary.

Term	Description
ADaM	Analysis Data Model
AI	Artificial intelligence
API	Application programming interface
BC	Biomedical concept; a standards-agnostic unit of knowledge defining what is observed, with the semantics and detailed metadata needed to generate CRFs, Define-XML, DTAs, and other study artifacts
CDISC	Clinical Data Interchange Standards Consortium
CDASH	Clinical Data Acquisition Standards Harmonization, a CDISC foundational standard that provides a universal, consistent way to collect raw data at the point of origin in clinical trials and research studies
CDASHIG	CDASH Implementation Guide; provides practical instructions and technical metadata tables for collecting consistent clinical trial and research data at the point of origin
Concept surrogate	A placeholder for an assessment lacking a published BC/specialization; carries an NCIt reference but no VLM
CORE	CDISC Open Rules Engine
CRF	Case report form
CRO	Contract (or clinical) research organization
CSV	comma-separated values
CT	(CDISC) Controlled Terminology
Dataset-JSON	Data exchange standard for sharing tabular data using JSON; designed to meet a wide range of data exchange scenarios, including regulatory submissions and API-based data exchange
DDE	(CDISC 360i) Data Definition Engine (project)
DDF	Digital data flow
DDS	Data definition specification, an intermediate operational specification layer between the USDM and the Define-XML; not yet fully standardized
Define-XML	CDISC standard for transmitting metadata that describes any tabular dataset structure
Domain	A collection of observations with a topic-specific commonality
DSS	Dataset specialization(s), the implementation layer for a BC; SDTM DSS provide a ready-to-use building block to instantiate a dataset, including value-level metadata
DTA	Data transfer agreement
eCTD	Electronic Common Technical Document (format)
EDC	Electronic data capture
FDA	(US) Food and Drug Administration
Foundational standards	CDISC standards (SDTM, ADaM, SEND, CDASH, and QRS) focusing on the core principles for defining data standards; these include models, domains and specifications for data representation
HTML	Hypertext markup language
Java	A high-level, object-oriented, memory-safe programming language known for its "write once, run anywhere" (WORA) capability
JSON	JavaScript Object Notation, a lightweight, text-based data-interchange format designed to be human readable and easy for machines to parse and generate

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Term	Description
JSONata	JSON query and transformation language
LOINC	Logical Observation Identifiers Names and Codes; international standard for identifying health observations, measurements, and documents
MDR	Metadata repository
MedDRA	Medical Dictionary for Regulatory Activities
NCI code	National Cancer Institute code (also C-code); an alphanumeric identifier that uniquely identifies a specific concept, term, or codelist within standardized clinical trial data
PMDA	(Japan) Pharmaceuticals and Medical Devices Agency
Python	An interpreted, object-oriented, high-level programming language with dynamic semantics
QRS	Questionnaires, Ratings, and Scales; instruments or assessments used in clinical research to provide a qualitative or quantitative assessment of a clinical concept or task-based observation
R	A free, open-source programming language and software environment explicitly designed for statistical computing, data analysis, and graphical visualization
REST API	Representational state transfer application programming interface, an architectural design pattern that enables different software applications to communicate over the internet
SAS	A fourth-generation programming language used primarily for data management, advanced analytics, and statistical analysis
SAS XPT	Binary transport file format
SDTM	Study Data Tabulation Model
SDTMIG	SDTM Implementation Guide
SoA	Schedule of activities
Study design solution	Custom-built or commercially available software that aids in digitally capturing the study design, which can be represented as a clinical study protocol and can utilize/export the design in USDM format
TDM	(SDTM) Trial Design Model
USDM	Unified Study Definitions Model
USDM-IG	USDM Implementation Guide
USDM-HB1	USDM Implementation Handbook 1
USDM-HB2	USDM Implementation Handbook 2
USDM-HB3	USDM Implementation Handbook 3
VLM	Value-level metadata; describes a variable's definition under a specific condition (WhereClause), e.g., a particular test code
WhereClause	Part of Define-XML; the condition attached to value-level metadata (e.g., VSTESTCD EQ "SYSBP"), possibly composite
WHODrug	World Health Organization Drug Dictionary; a global, standardized dictionary of medicinal products
XML	Extensible markup language, a flexible text format used to store and transport structured data

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Appendix D: References

1. Jansen L. Working with biomedical concepts and SDTM dataset specializations for Define-XML v2.1 using SAS openCST. PharmaSUG 2023, Paper SS-140. 2023. Accessed November 4, 2026. https://pharmasug.org/proceedings/2023/SS/PharmaSUG-2023-SS-140.pdf

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved

Appendix E: Representations and Warranties, Limitations of Liability, and Disclaimers

CDISC Patent Disclaimers

It is possible that implementation of and compliance with this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any claim or of any patent rights in connection therewith. CDISC, including the CDISC Board of Directors, shall not be responsible for identifying patent claims for which a license may be required in order to implement this standard or for conducting inquiries into the legal validity or scope of those patents or patent claims that are brought to its attention.

Representations and Warranties

"CDISC grants open public use of this User Guide (or Final Standards) under CDISC's copyright."

Each Participant in the development of this standard shall be deemed to represent, warrant, and covenant, at the time of a Contribution by such Participant (or by its Representative), that to the best of its knowledge and ability: (a) it holds or has the right to grant all relevant licenses to any of its Contributions in all jurisdictions or territories in which it holds relevant intellectual property rights; (b) there are no limits to the Participant's ability to make the grants, acknowledgments, and agreements herein; and (c) the Contribution does not subject any Contribution, Draft Standard, Final Standard, or implementations thereof, in whole or in part, to licensing obligations with additional restrictions or requirements inconsistent with those set forth in this Policy, or that would require any such Contribution, Final Standard, or implementation, in whole or in part, to be either: (i) disclosed or distributed in source code form; (ii) licensed for the purpose of making derivative works (other than as set forth in Section 4.2 of the CDISC Intellectual Property Policy ("the Policy")); or (iii) distributed at no charge, except as set forth in Sections 3, 5.1, and 4.2 of the Policy. If a Participant has knowledge that a Contribution made by any Participant or any other party may subject any Contribution, Draft Standard, Final Standard, or implementation, in whole or in part, to one or more of the licensing obligations listed in Section 9.3, such Participant shall give prompt notice of the same to the CDISC President who shall promptly notify all Participants.

No Other Warranties/Disclaimers. ALL PARTICIPANTS ACKNOWLEDGE THAT, EXCEPT AS PROVIDED UNDER SECTION 9.3 OF THE CDISC INTELLECTUAL PROPERTY POLICY, ALL DRAFT STANDARDS AND FINAL STANDARDS, AND ALL CONTRIBUTIONS TO FINAL STANDARDS AND DRAFT STANDARDS, ARE PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE, AND THE PARTICIPANTS, REPRESENTATIVES, THE CDISC PRESIDENT, THE CDISC BOARD OF DIRECTORS, AND CDISC EXPRESSLY DISCLAIM ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR OR INTENDED PURPOSE, OR ANY OTHER WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, FINAL STANDARDS OR DRAFT STANDARDS, OR CONTRIBUTION.

Limitation of Liability

IN NO EVENT WILL CDISC OR ANY OF ITS CONSTITUENT PARTS (INCLUDING, BUT NOT LIMITED TO, THE CDISC BOARD OF DIRECTORS, THE CDISC PRESIDENT, CDISC STAFF, AND CDISC MEMBERS) BE LIABLE TO ANY OTHER PERSON OR ENTITY FOR ANY LOSS OF PROFITS, LOSS OF USE, DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, OR SPECIAL DAMAGES, WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS POLICY OR ANY RELATED AGREEMENT, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Note: The CDISC Intellectual Property Policy can be found at http://www.cdisc.org/system/files/all/article/application/pdf/cdisc_20ip_20policy_final.pdf

© 2026 Clinical Data Interchange Standards Consortium, Inc. All rights reserved