**RegistryDatabase: Developer Documentation**

Last Updated: May 11, 2025

**1. Introduction**

* Database Name: RegistryDatabase
* Database Type: Microsoft SQL Server
* Primary Purpose: The RegistryDatabase is designed to allow users to create, manage, and document data Specifications. Users can define a new Specification and then select and customize elements from pre-defined "model" tables (representing a Core Invoice Model and various Extension Components) to build up their specific data requirements.

**2. Database Schema Overview**

The database schema can be broadly categorized into two types of tables:

* Model Tables: These tables store the master list of data elements, components, and their attributes that users can choose from. The data for these tables is typically pre-populated (as indicated by the "RegistryModel Data" excel file).
  + CoreInvoiceModel
  + ExtensionComponentsModelHeader
  + ExtensionComponentModelElements
* Specification Tables: These tables store the user-defined Specifications. Each Specification consists of identifying information and selected elements from the model tables, potentially with user-specific customizations or notes.
  + SpecificationIdentifyingInformation
  + SpecificationCore
  + SpecificationExtensionComponents

**3. Model Tables**

These tables provide the foundational data elements that users can select when creating a Specification.

**3.1.** CoreInvoiceModel

* Purpose: Stores the elements of a standard "Core Invoice" model. Users select from these elements when defining the core part of their Specification.
* Primary Key: ID (varchar(10))
* Key Columns:
  + ID: Unique identifier for the Core Invoice Model element – the Business Term ID e.g. BT-1, or BG-1.
  + BusinessTerm: The name or term for the metadata element (e.g., "Invoice Number", "Issue Date").
  + Level: Hierarchical level of the element.
  + Cardinality: Defines how many times the element can appear (e.g., "1..1", "0..n").
  + SemanticDescription: Explanation of the element's meaning.
  + DataType: The data type of the element (e.g., "Text", "Date").
  + ParentID: Identifier of the parent element in a hierarchy.

**3.2.** ExtensionComponentsModelHeader

* Purpose: Stores header or grouping information for "Extension Components." An Extension Component might represent a collection of related data elements that can be added to a Specification.
* Primary Key: ID (varchar(10))
* Key Columns:
  + ID: Unique identifier for the Extension Component – the Business Term ID e.g. BT-1, XT-1, BG-1 or XG-1.
  + Name: Name of the Extension Component element.
  + Status: Status of the component (e.g., "Active", "Deprecated").
  + ECLink: A hyperlink to the Extension Component in Github.

**3.3.** ExtensionComponentModelElements

* Purpose: Stores the detailed individual data elements that belong to each "Extension Component" defined in ExtensionComponentsModelHeader.
* Primary Key: EntityID (int, IDENTITY) – this is an autonumber
* Foreign Keys:
  + ExtensionComponentID REFERENCES ExtensionComponentsModelHeader(ID)
* Key Columns:
  + ExtensionComponentID: Links the element to its parent Extension Component in ExtensionComponentsModelHeader.
  + BusinessTermID: A specific identifier for the business term within the context of an Extension Component.
  + BusinessTerm: The name or term for the data element.
  + Level, Cardinality, SemanticDescription, DataType, ParentID: Similar to CoreInvoiceModel but specific to extension elements.
  + ExtensionType: Category or type of extension.
* Important Constraints:
  + A UNIQUE constraint UQ\_ExtensionComponentModelElements\_Component\_BusinessTerm exists on the combination of (ExtensionComponentID, BusinessTermID). This ensures that each business term is unique within a given Extension Component and is crucial for reliably referencing these elements from the SpecificationExtensionComponents table.

**4. Specification Tables**

These tables store the Specifications created by users.

**4.1.** SpecificationIdentifyingInformation

* Purpose: This is the central table for a user-defined Specification. It stores the main metadata and identifying details for each Specification created by a user.
* Primary Key: IdentityID (int, IDENTITY)
* Key Columns:
  + IdentityID: Unique identifier for the user-defined Specification.
  + SpecificationIdentifier: A user-friendly or official identifier for the Specification.
  + SpecificationName: The name of the Specification.
  + Sector, SubSector: Industry sector/sub-sector the Specification applies to.
  + Purpose: The reason or goal of the Specification.
  + SpecificationVersion: Version number of the Specification.
  + ContactInformation: Contact details related to the Specification.
  + DateOfImplementation: Planned or actual implementation date.
  + CoreVersion: Version of the core model used.

**4.2.** SpecificationCore

* Purpose: Links a user-defined Specification (from SpecificationIdentifyingInformation) to selected elements from the CoreInvoiceModel. It stores how each selected core model element is used or modified within that specific Specification.
* Primary Key: EntityID (int, IDENTITY) – an autonumber
* Foreign Keys:
  + IdentityID REFERENCES SpecificationIdentifyingInformation(IdentityID)
  + BusinessTermID REFERENCES CoreInvoiceModel(ID)
* Key Columns:
  + IdentityID: Identifies the user Specification.
  + BusinessTermID: Identifies the selected element from CoreInvoiceModel.
  + Cardinality: Defines how many times the element can appear (e.g., "1..1", "0..n") (may override model's default).
  + UsageNote: Notes on how the element is used in this Specification.
  + TypeOfChange: Describes any modification from the Core Invoice Model.

**4.3.** SpecificationExtensionComponents

* Purpose: Links a user-defined Specification to selected elements from the ExtensionComponentModelElements table. This table details which extension elements are included in a Specification and any specific usage notes or justifications.
* Primary Key: EntityID (int, IDENTITY)
* Foreign Keys:
  + IdentityID REFERENCES SpecificationIdentifyingInformation(IdentityID)
  + (ExtensionComponentID, BusinessTermID) REFERENCES ExtensionComponentModelElements(ExtensionComponentID, BusinessTermID) (Constraint Name: FK\_SpecExtComponents\_To\_ExtCompElements)
* Key Columns:
  + IdentityID: Identifies the user Specification. – refers to a specific SpecificationIndetifyingInformation Table record
  + ExtensionComponentID: Identifies the specific Extension Component from which the element is chosen.
  + BusinessTermID: Identifies the specific Business Term element within that Extension Component.
  + Cardinality: Defines how many times the element can appear (e.g., "1..1", "0..n"). May override the Cardinality of the element in the Extension Component.
  + UsageNote: Notes on how the extension element is used.
  + Justification: Reason for including/modifying this extension element.
  + TypeOfExtension: Type of extension applied.
* Important Notes on Relationship:
  + The ExtensionComponentID column in this table has been explicitly set to NOT NULL to ensure it always has a value when linking to ExtensionComponentModelElements.
  + The foreign key FK\_SpecExtComponents\_To\_ExtCompElements is a composite key, linking to the unique combination of ExtensionComponentID and BusinessTermID in the ExtensionComponentModelElements table. This relationship is enforced by the unique constraint UQ\_ExtensionComponentModelElements\_Component\_BusinessTerm on the ExtensionComponentModelElements table.

**5. Key Relationships and Data Flow**

The general workflow for creating a Specification is as follows:

1. Create Specification Header: A user initiates a new Specification by adding a record to the SpecificationIdentifyingInformation table. This record receives a unique IdentityID.
2. Select Core Model Elements: The user then selects relevant data elements from the CoreInvoiceModel table. For each selected element, a record is added to the SpecificationCore table, linking back to the IdentityID of the Specification and the ID (as BusinessTermID) of the chosen CoreInvoiceModel element. This records which CoreInvoiceModel element was included in the Specification. Also it records changes to details like cardinality or usage and its conformance type.
3. Select Extension Component Elements: Similarly, the user selects relevant data elements from the available ExtensionComponentModelElements. For each selected extension element, a record is added to the SpecificationExtensionComponents table. This record links to:
   * The IdentityID of the Specification.
   * The ExtensionComponentID of the chosen Extension Component.
   * The BusinessTermID of the chosen element within that component.

Specificatiom-specific details like cardinality, justification, and usage notes are recorded here.

A screenshot of a computer

AI-generated content may be incorrect.

**6. Important Constraints and Modifications for Data Integrity**

Several constraints and schema modifications are critical for maintaining data integrity, particularly concerning the relationship between SpecificationExtensionComponents and ExtensionComponentModelElements:

* UQ\_ExtensionComponentModelElements\_Component\_BusinessTerm on ExtensionComponentModelElements: This UNIQUE constraint on (ExtensionComponentID, BusinessTermID) in the ExtensionComponentModelElements table is essential. It ensures that the combination of an Extension Component and a business term within it is unique, allowing it to be reliably targeted by a foreign key.
* ExtensionComponentID in SpecificationExtensionComponents is NOT NULL: This column was explicitly altered to NOT NULL. This change is vital because ExtensionComponentID is part of the composite foreign key referencing ExtensionComponentModelElements. A NULL value would prevent a valid foreign key relationship.
* FK\_SpecExtComponents\_To\_ExtCompElements on SpecificationExtensionComponents: This composite FOREIGN KEY ([ExtensionComponentID], [BusinessTermID]) REFERENCES [dbo].[ExtensionComponentModelElements] ([ExtensionComponentID], [BusinessTermID]) establishes the formal link between a user's Specification of an extension element and the definition of that element in the model. It ensures that only valid, existing extension elements can be added to a Specification.

These elements (the UNIQUE constraint, the NOT NULL column, and the composite FOREIGN KEY) work together to enforce referential integrity and ensure consistency in how Extension Components are specified.

**7. Data Source for Model Tables**

As indicated, the initial data for the "model" tables (CoreInvoiceModel, ExtensionComponentsModelHeader, ExtensionComponentModelElements) is expected to be populated from external sources, such as the provided CSV files (RegistryModelData.xls - CoreInvoiceModel.csv, etc.).

**8. Conclusion**

The RegistryDatabase provides a structured way for users to define detailed data Specifications by leveraging and customizing elements from established core and extension models. The relationships, particularly the carefully constructed link between SpecificationExtensionComponents and ExtensionComponentModelElements, are key to its functionality and data integrity. Developers working with this database should pay close attention to these relationships and the constraints that enforce them.