Getting Started

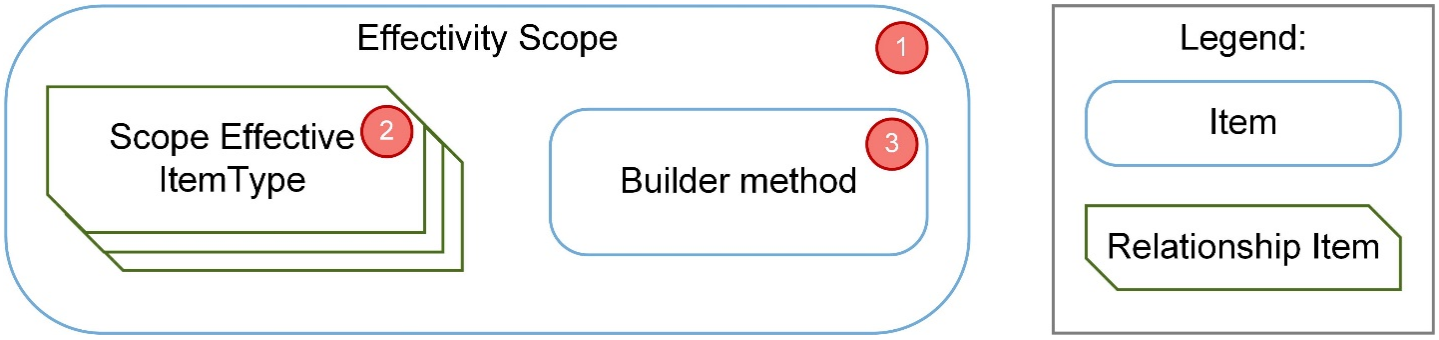
Effectivity Services is used for resolving structures with effectivity conditions on Relationship Items for specified criteria.

The Aras Platform provides the following Effectivity Services features:

- Effectivity Services Data Model built with new ItemTypes, RelationshipTypes, Methods and more to enable customization and/or extension to support a custom flow.
- Effectivity Resolution Engine to determine which items are effective in a structure for a specific configuration.

Effectivity Services Data Model Concept Overview

The Effectivity Services data model concept contains the following main structural blocks:



1. **Effectivity Scope** is an item of the ItemType “effs_scope” used for defining the context for effectivity resolution. An Effectivity Scope item requires a reference to the “Builder method” and may have “Scope Effective ItemType” relations to the RelationshipTypes where effectivity is managed.
2. **Effectivity Scope ItemType** is an item of the RelationshipType “effs_scope_itemtype” which creates a relationship between an Effectivity Scope item and an ItemType (“RelationshipType” ItemType). When this relationship is established, the Effectivity Services Core automatically creates a new null (No Related) RelationshipType for the specified ItemType and names it according to the template “effs_%ITEMTYPE%_expression”, where %ITEMTYPE% is the ItemType name.
   The RelationshipType “effs_%ITEMTYPE%_expression” is added as a poly source to the

polymorphic ItemType “effs_expression” to subscribe to the Effectivity Resolution Engine. This RelationshipType retains Effectivity Expressions.

3. **Builder Method** is an item of the ItemType “Method”. A reference to it is stored in the “builder_method” property associated with the Effectivity Scope item. The Effectivity Resolution Engine calls this method to construct a Scope object, which serves as the base for effectivity resolution. For more information about the builder method, refer to section 5.2.1 Implementing the Scope Builder Method.

Important

Effectivity Scope and Scope Object are different. The Effectivity Scope is an item of the ItemType “effs_scope”. The Scope Object is an instance of the class “Aras.Server.Core.Configurator.Scope,” which is the result of Builder method execution.

Effectivity Services Data Model Technical Overview

This section contains a technical description of the Effectivity Services Data Model.

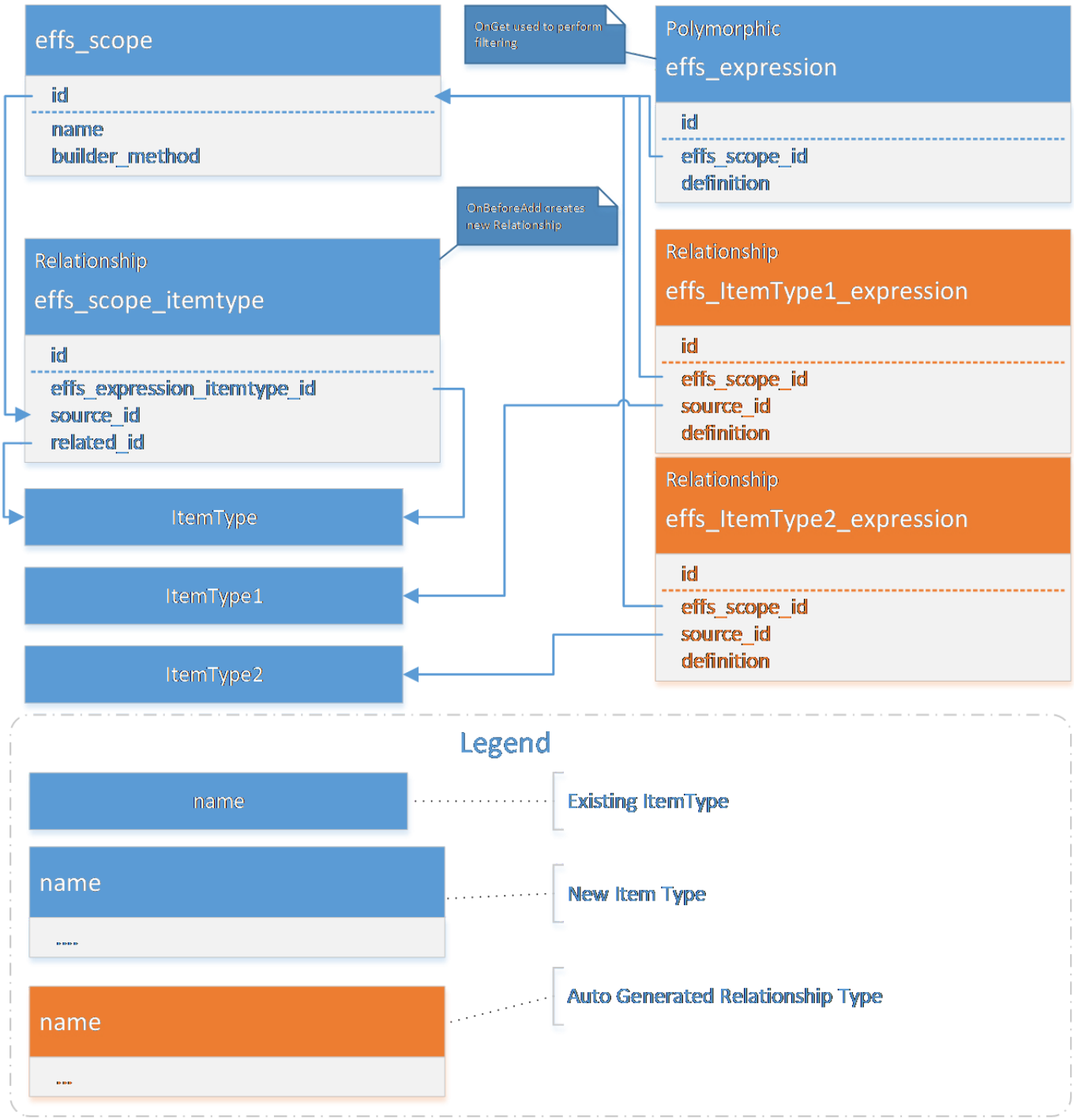
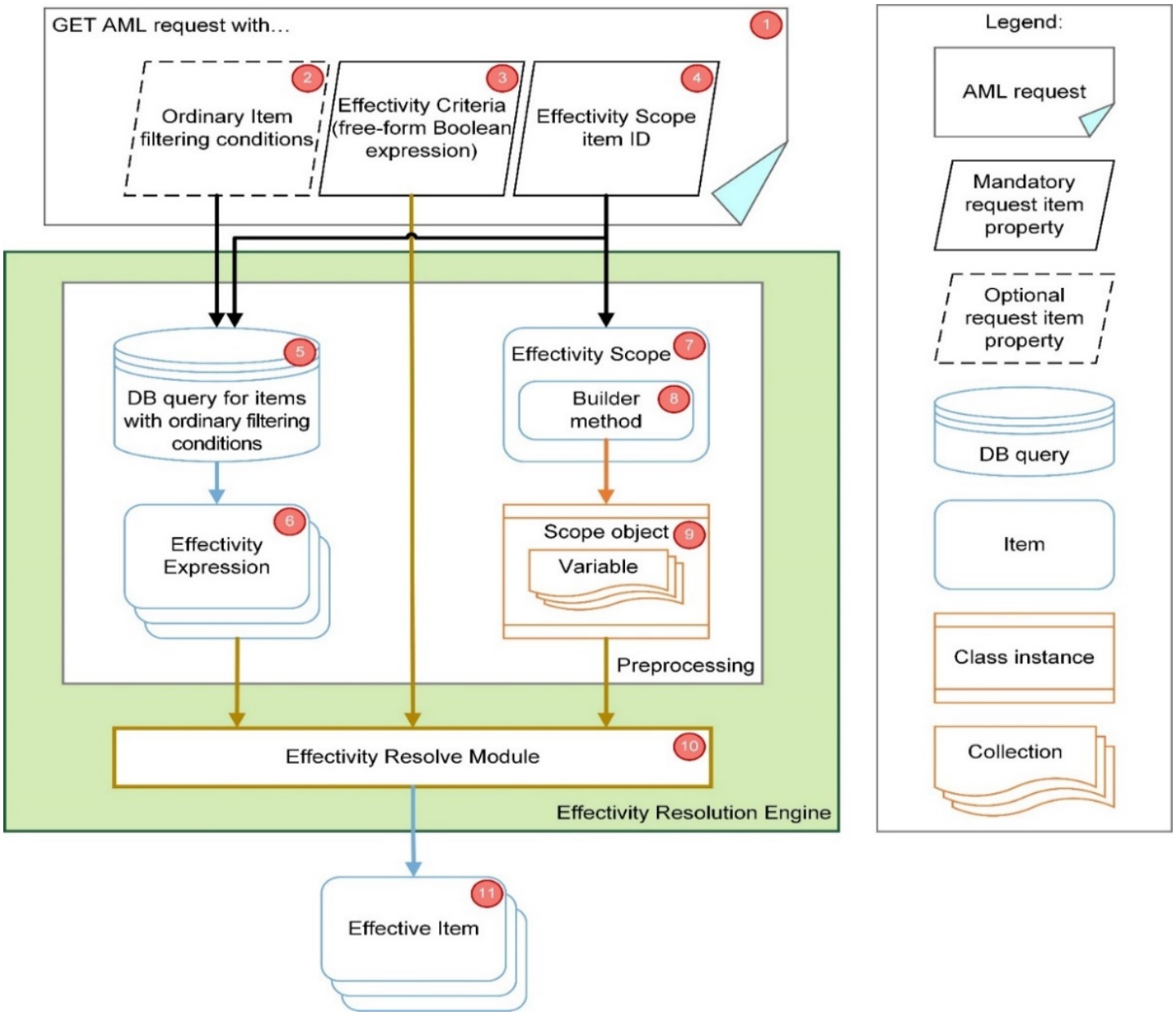


Table 2: Effectivity Services Data Model

Item Type	Property	Description
ItemType		Standard ItemTypeItemType, holder of allItemTypesin Aras
ItemType1, ItemType2		RelationshipItemTypesthat ar managed byeffectivity.
effs_scope,effs_scope_itemtype,effs_expression		Effectivity Data Model
effs_scope		Container for scope object de
	name	Name of Scope Item
	builder_method	Link to custom builder method construct Scope object for cu
effs_scope_itemtype		Relationship between Scope (source_id) and target Item Type(related_id) onBeforeAdd event auto-gene related relationship with name effs_#ItemType#_expre Just one Item can be created pair scope and item type.
	effs_expression_itemtype_id	itemTypeid,effs_#ItemType#_
	source_id	effs_scopeid
	related_id	itemTypeid, target relationshi Type
effs_expression		Polymorphic ItemType withon for Effective Itemsfiltering
	effs_scope_id	effs_scopeid
	definition	Container for expression
effs_#ItemType#_expression		Poly source foreffs_expressio related relationship to target I
	effs_scope_id	effs_scopeid
	source_id	item id for target Item Type
	definition	Container for expression

Effectivity Services Process Flow Overview

The following diagram shows the process flow for Effectivity Services.



1. **An AML request with the action “get”,** where there are mandatory and optional request item properties (nodes) to apply effectivity resolution on a requested structure.

> **Important**
>
> The Effectivity Resolution Engine intercepts only “GET” AML requests.

The current implementation of the Effectivity Resolution Engine can intercept “GET” AML requests to ItemTypes which are poly sources of the polymorphic ItemType “effs_expression”. Therefore, it can process items of poly source ItemTypes of the polymorphic ItemType “effs_expression” to determine effective items.

2. **Ordinary item filtering conditions** in an AML request are used as criteria to limit the items requested from a database (DB) in the preprocessing stage of effectivity resolution processing. The GetItem internal module uses them to request items from the DB to fulfill specified filtering conditions. For example, these criteria may be item id, a list of item ids (known as the “idlist” attribute on a request Item node), or item property tags in an AML request. Ordinary item filtering conditions are optional and provided as needed.

3. **Effectivity Criteria** is a condition (e.g., Model = Model-X and Production Date = 2021/06/28) written in the Free-Form Boolean expression. For more information, refer to section 4 Free-Form Boolean Expression Language. The criteria are used to evaluate each effectivity expression in a requested structure to determine effective items. An item is effective within a structure if the evaluation of its corresponding Effectivity Expression(s) against Effectivity Criteria produces the logical truth.

   The representation of the “Effectivity Criteria” mandatory field in an AML request is the node “definition” matching the XPath “Item/definition | Item/and/definition” and containing a free-form Boolean expression in the XML-encoded form.

   > **Important**
   >
   > The node “definition” is not passed to the internal GetItem module. Therefore, it will not affect the items requested from a database.

4. **Effectivity Scope item ID** is the unique identifier (ID) of an Effectivity Scope Item used as a context for effectivity resolution.
   The representation of the “Effectivity Scope item ID” mandatory field in an AML request is the node “effs_scope_id” matching the XPath “Item/effs_scope_id | Item/and/effs_scope_id” and containing 32-hex GUID.

   > **Important**

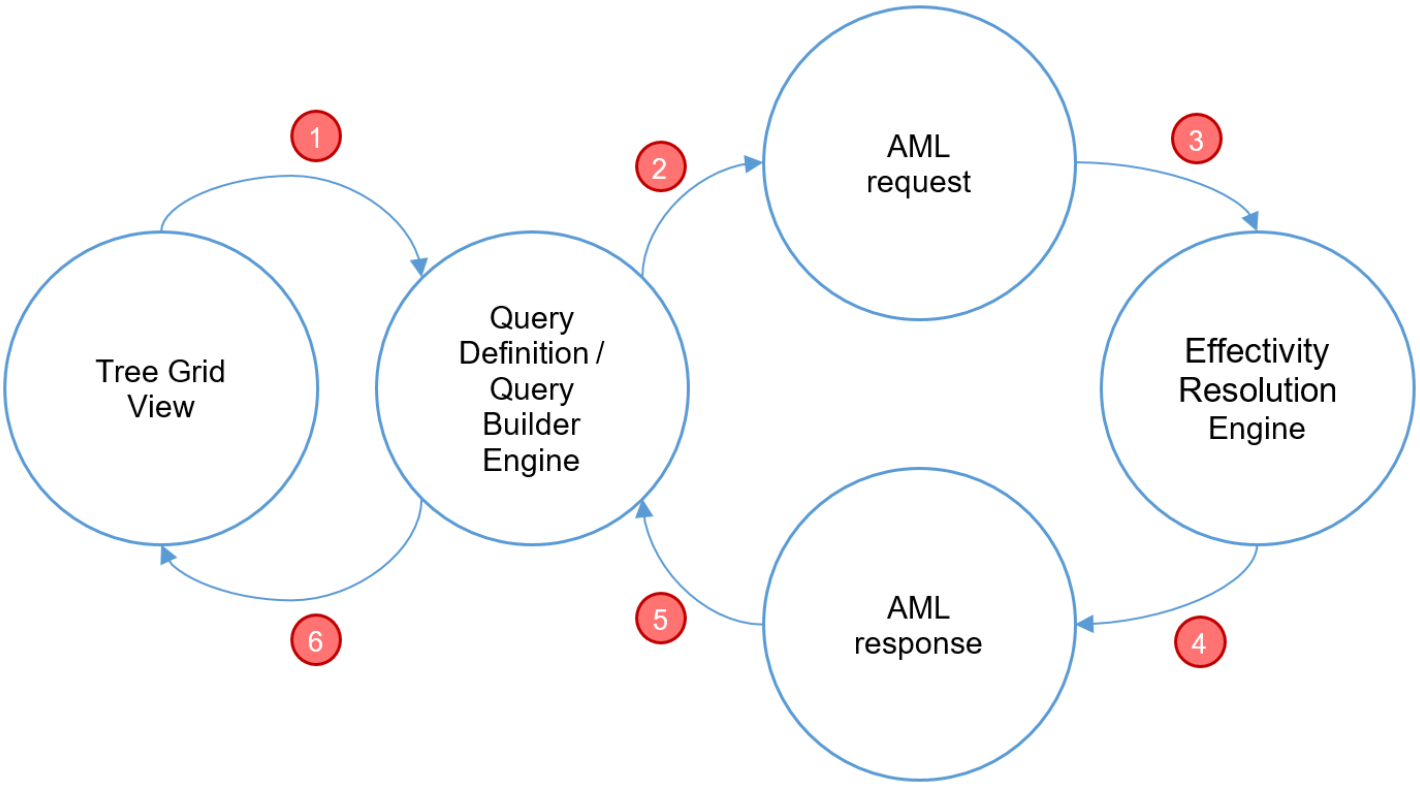
The node “effs_scope_id” is passed to the internal GetItem module as part of querying the database.

5. **DB query for items** with ordinary filtering conditions is an action on the preprocessing stage of effectivity resolution. The internal GetItem module requests items from a DB with optional item filtering conditions and the mandatory property “effs_scope_id”. A response has items with effectivity expressions for evaluation with Effectivity Criteria.
6. **Effectivity Expression items** are the result of querying a DB for items, which fulfill specified optional item filtering conditions and the mandatory property “effs_scope_id”.
7. **Effectivity Scope** is an item of the ItemType “effs_scope” requested from a DB using the incoming mandatory request property “effs_scope_id”. This is to get a builder method referenced in the property “builder_method” of an Effectivity Scope item.
8. **Builder Method** is an item of the ItemType “Method”. The preprocessing stage of Effectivity Resolution extracts the last generation of the method code using a reference in an Effectivity Scope item, compiles the code and calls it with the single argument “Effectivity Scope Id” to return a properly constructed Scope object.
9. **Scope object** is a properly constructed instance of the class “Aras.Server.Core.Configurator.Scope”, which has a list of Effectivity Variables, which will take a part in Effectivity Resolution.
10. **Effectivity Resolve Module** takes Effectivity Expression items, Effectivity Criteria, and a Scope object to evaluate each Effectivity Expression item against Effectivity Criteria. If this evaluation produces the logical truth, the item goes to the resulting structure.
11. **Effective items** are the result of the effectivity resolution process flow. The final structure contains only the items determined to be valid for the specified effectivity criteria.

Process Flow Overview of Effectivity Services with Query Builder and Tree Grid View

Combining Effectivity Services with the Query Builder and Tree Grid View applications is one of the native ways to visualize the results of the effectivity resolution process in the user interface. The intended use of the Query Builder application is to create Query Definitions which are reusable AML requests. For more information about using Query Builder, refer to the Aras Innovator 40 – Query Builder Guide. The Tree Grid View application enables you to build a visual data structure. For more information about Tree Grid View, refer to the Aras Innovator 40 – Tree Grid View Administrator Guide. Effectivity Services acts as an interlayer in the native Query Definition process flow. The Effectivity Resolution Engine performs additional processing on the items requested by Query

Definition. **Figure 4** illustrates the process flow for Effectivity Services combined with Query Builder and Tree Grid View.



1. The configured **Tree Grid View** uses its associated Query Definition to query data to display. For information on how to configure the Tree Grid View, refer to section 6.3.2 Creating a Tree Grid View to Display Effective Items.
2. The **Query Builder Engine** processes the received Query Definition to retrieve data from a database (DB) and return it. To complete this task, the Query Builder Engine prepares and sends the AML request internally. For information on how to configure the Query Definition, refer to section 6.3.1 Creating a Query Definition to Filter by Effectivity.
3. The **Effectivity Resolution Engine** receives the AML request from the Query Builder Engine, gets the necessary data from a DB by applying the request, and filters data according to the Effectivity Criteria.
4. The **Effectivity Resolution Engine** prepares the AML response containing the effective items and sends it back to the Query Builder Engine.
5. The **Query Builder Engine** receives and processes the requested data according to the requirements of the regular query execution flow.

6. The **Query Builder Engine** sends the items structure to the Tree Grid View to display.