STIBO SYSTEMS
MASTER DATA MANAGEMENT

USER GUIDE

Performance Recommendations

Release 2023.4 – December 2023

Table of Contents

Table of Contents 2

Performance Recommendations 6

Attribute and Attribute Group Recommendations 7

Attribute Groups 7

Attributes 8

Calculated Attribute Recommendations 12

Considerations and Limitations 13

Recommendations 14

LOV Filtering Recommendations 15

Recommendations 16

Manually Sorted Recommendations 17

Recommendations 18

Background Process Queue Recommendations 19

Recommendations 19

Base Setup Recommendations 21

Asset Recommendations 22

Storage Location 22

Dimension Dependency 22

Import and Export 23

Classification Recommendations 24

Recommendations 25

Data Model Recommendations 26

Dimension and Context Recommendations 28

Recommendations 30

Global Count of Object and Attribute Recommendations 32

Reference Recommendations 35

Recommendations 35

Business Rule Recommendations 36

Extending STEP Functionality 36

Business Rule Analysis 38

Using Business Rules in STEP 38

Test & Time Business Rule 40

Business Rules Statistics 40

Admin Portal Business Rule Activity Dashboard 41

Admin Portal Business Rule Tracing 42

Business Rule Elements to Use 44

Use Exception Handling 44

Use Logging Carefully 45

Use Arrays, Not Multiple Read Calls 46

In-Memory 47

Consider Using an Extension 47

Business Rule Elements to Avoid 49

Avoid Large Transactions 49

Avoid Large Libraries 49

Avoid Many Inclusions of a Library 50

Avoid Infinite Loops 50

Avoid the 'getChildren' Function with Many Nodes 50

Avoid Updating Data via Business Conditions 51

Event Processor and Event Queue Recommendations **52**

Run Without Warnings and Errors 52

Optimize STEP Setup for Performance 52

Analyze Asynchronous Processing 53

Legacy - Queues and Queue Size 55

Consider In-Memory for Event Processors 56

Export Recommendations **57**

Export Elements to Use **58**

Optimize Object Type Triggering Definitions 58

Optimize Attribute and Reference Triggering Definitions 58

Optimize Event Filter and Event Generator Triggering Definitions 58

Use Multiple Dedicated OIEPs 59

Use Multithreading 59

Optimize the Batch Size 60

Use Cross-Context Exports 61

Use Event-Based Exports Over Static Exports 61

Legacy - Use Separate Queues for Important OIEPs 62

Consider In-Memory for Exports 63

Optimize STEP Setup for Export Performance 63

Export Elements to Limit **64**

Limit Event-Based OIEPs 64

Limit the Volume of Exported Data 64

Limit Unnecessary Data 64

Limit Multiple Output Templates 65

Avoid Complex Export Privileges 65

Import Recommendations **66**

Import Elements to Use **67**

Use Term Lists for Price Data 67

Use Business Rules Designed for Import Performance 67

Use Workflow Initiations Designed for Import Performance 67

Use Approvals Designed for Import Performance 68

Use Event-Based Exports Designed for Import Performance 68

Legacy - Use Separate Queues for Important IIEPs 68

Optimize STEP Setup for Import Performance 69

Import Elements to Avoid **71**

Avoid Typical Import Errors 71

Avoid Missing Reference Targets 71

Avoid Forward Declarations 72

Avoid Multiple Updates of Same Object 72

Avoid Unnecessary Business Rules on Import 72

Avoid Optimistic Locking in Business Rules on Import 73

Avoid Complex Privileges on Import 73

Importing for Migration **74**

Matching and Linking Recommendations **75**

Optimize Match Codes and Matching Algorithm 75

Statistics on Match Codes Generation 75

Statistics on Running the Matching Algorithm 76

Limit Attributes Promoted to Golden Records 77

Approve Golden Records Outside of Matching and Linking 77

Avoid Multi-Context Survivorship Rules 77

Optimistic and Pessimistic Locking Recommendations 79

'Reference Target Lock Policy' Parameter 80

Strict or Relaxed 80

Optimistic Locking - Log Level Setting 81

Optimistic Locking - Analyzing Failures 81

Optimistic and Pessimistic Locking Recommendations 83

Privilege Recommendations 85

Privilege Configurations 85

Profiling Recommendations 86

Recommendations 87

Revision Control Recommendations 89

Setting the Revision Threshold 89

Maintaining Object Revisions 89

Maintaining Integration Endpoints Revisions 90

Scheduled Process Recommendations 91

Recommendations 91

Create an Excel Overview of Scheduled Processes 92

Analyzing and Optimizing Scheduled Processes 93

Analyzing and Optimizing Long-Running Processes 94

Search Elements to Use 98

Basic Searches 98

Use Specific Search Criteria 99

Use Object Super Types 99

Searches with Wildcards 100

Optimize Search Below with a Configuration Property 100

Optimize Combined Search Below and Value Search 101

In-Memory 101

Search Elements to Avoid 102

Avoid Inherited Values and Regular Expressions 102

Avoid Root Hierarchies in Search Below 102

File Hygiene Recommendations 103

Clean Up Import Files 103

Identify the Causes 103

Remove Import Files 104

Create a Background Processes Maintenance Plan 105

IEP Auto Delete Settings 105

Background Process Auto Delete Settings 106

Web UI Configuration Recommendations 109

Web UI Component Report 109

Use Multiple Web UIs 110

Use Small Dedicated Web UI Screens 110

Avoid Using Images in Multi-Select Web UI Screens 112

Use 'Lazy' Loading for Web UI Screens 112

Use Type Ahead for LOVs in Web UI Screens 113

Correctly Configure Status Selectors in Web UI 114

Privileges in Web UI configuration 116

Consider In-Memory for Web UI Screens ..116

Performance Recommendations

This section describes the data gathering methodologies from a technical infrastructure viewpoint for the clean up of the STEP servers.

- Attribute and Attribute Group Recommendations
- Background Process Queue Recommendations
- Base Setup Recommendations
- Business Rule Recommendations
- Export Recommendations
- Event Processor and Event Queue Recommendations
- Import Recommendations
- Matching and Linking Recommendations
- Optimistic and Pessimistic Locking Recommendations
- Privilege Recommendations
- Profiling Recommendations
- Revision Control Recommendations
- Scheduled Process Recommendations
- File Hygiene Recommendations
- Web UI Configuration Recommendations

Attribute and Attribute Group Recommendations

This is one of the data gathering methodologies and recommendations for functional performance improvement. The full list is defined in the **Performance Recommendations** topic.

Configured attributes are:

- Specification or description
- Of a certain value type (text, list of value, number, date, etc.)
- Multi-valued or single valued
- Associated with a unit of measure or not.

While none of these configurations independently have a direct impact on performance, some attribute group and attribute configurations can influence performance.

Note: A review of the configured attribute groups and attributes is a good start in analyzing the configuration. Export the attribute groups to a STEPXML file and the attributes to an Excel file for review.

Attribute Groups

The following parameters can affect performance and apply to attribute groups:

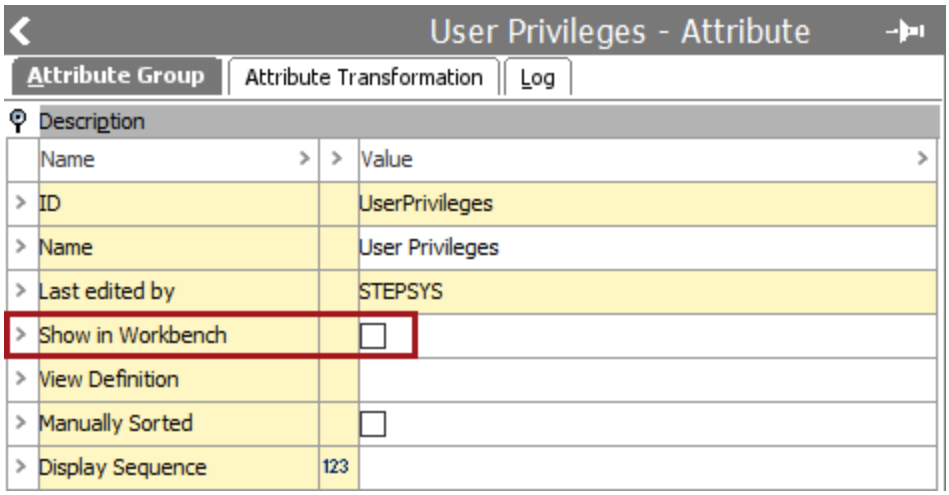
- Show in Workbench
- Manually Sorted

Show in Workbench

In System Setup, the 'Show in Workbench' parameter on an attribute allows the groups to be included or excluded from the workbench display. Showing many attribute values in the workbench requires additional processing power, and can become expensive in terms of performance, especially for the values of calculated attributes.

To review the attributes being displayed in the workbench, export all attribute groups as STEPXML and in the output file search for **ShowInWorkbench ="true"**. Determine if any of the attributes being displayed should be hidden.

Disable the 'Show in Workbench' option to prevent the attribute group from being displayed.



For better performance, set the 'Show in Workbench' parameter as enabled only for attribute groups that need to be displayed in the workbench.

Manually Sorted

'Manually Sorted' adds complexity and can cause performance issues. Refer to the **Manually Sorted Recommendations** topic for more information.

Attributes

The following parameters can affect performance and apply to attributes:

- Full Text Indexable
- Dimension Dependencies
- Large Lists of Values (LOVs)

Full Text Indexable

In System Setup, the 'Full Text Indexable' parameter can be set on an attribute with a Validation Base Type of 'text' so you can search for words within 'text' values. For example, it allows the user to search for objects based on a word, or set of words, in a sentence within an attribute value. Without the full text indexable option, you can still search for values.

When an attribute is made as Full Text Indexable, performance can be negatively affected. For more information, refer to the **Full Text Indexable Attributes** topic in the **System Setup** documentation.

Attribute | References | Attribute Transformation | Validity | Profile | Lo

Description

Name	Value
ID	Consumer Description
Name	Consumer Description
Last edited by	2018-01-25 10:43:29 by STEPSYS
Full Text Indexable	No
Externally Maintained	No
Completeness Score	
Hierarchical Filtering	None
Calculated	No
Type	Specification
Dimension Dependencies	Language;
Mandatory	Yes

To review the attributes with this option enabled, export all attributes as STEPXML and in the output file search for **FullTextIndexed="true"**. Determine if the parameter can be disabled.

For better performance, set the 'Full Text Indexable' parameter to 'Yes' only if needed.

Dimension Dependencies

Dimension Dependencies add complexity and cause performance issues, and should only be used on attributes when necessary.

To review the attributes with this option enabled, export all attributes as Excel and in the output file filter on **<Attribute Dimension dependencies>**.

Refer to the **Dimension and Context Recommendations** topic for more information.

Attribute | References | Attribute Transformation | Validity | Profile | Lo

Description

Name	Value
ID	Consumer Description
Name	Consumer Description
Last edited by	2018-01-25 10:43:29 by STEPSYS
Full Text Indexable	No
Externally Maintained	No
Completeness Score	
Hierarchical Filtering	None
Calculated	No
Type	Specification
Dimension Dependencies	Language;
Mandatory	Yes

Large Lists of Values (LOVs)

Special attention is required for attributes with large List Of Values (LOVs)—those with thousands of values. LOVs with many values can lead to performance degradation. Additionally, from a usability perspective, large LOVs should be avoided.

Use the Healthcheck in the Admin Portal to find LOVs with too many values. For more information, refer to the **Healthcheck** topic in the **Administration Portal** documentation.

Monitoring | Configuration | Thread Dump | Tools | Profiler | Healthcheck | Send Diagnostics | Localization

The objective behind the Healthcheck functionality is, to visualize information which have not previously been accessible to customers and hen as a customer the opportunity to intervene and rectify the current state before it potentially escalates to a real performance issue. Please note, functionality is designed to be run on a regular basis, which allows you to continuously monitor and thus avoid minor issues to evolve into more situation. Please find detailed information in the 'My Issues/Document Repository' area, in JIRA.

Server time: 04/08/2019

Available Tests

Test	Last Run	Execution Time (Last ...	Executed By	Detected Problems
Too many valid values for list of values.	Tue Apr 24 02:25:33 EDT 2018	1 sec	USERY	2

For better performance, split large LOVs into smaller LOVs, or use a text field instead of an LOV.

For more information, refer to the **LOV Filtering Recommendations** topic.

Calculated Attributes

Calculated attributes provide a lot of flexibility, but they also have a direct impact on downstream deliveries. Additionally, calculated attributes can degrade performance. For more information, refer to the **Calculated Attribute Recommendations** topic in this documentation.

Calculated Attribute Recommendations

This is one of the data gathering methodologies and recommendations for base setup improvement. The full list is defined in the **Base Setup Recommendations** topic.

Note: These recommendations are valid for systems without In-Memory. The performance costs are different with In-Memory. For more information, refer to the **In-Memory Database Component for STEP** section of the **Resource Materials** in online help.

Calculated attributes are a special type of attributes whose values are not written in the database, but are calculated on the fly. Calculated values are dynamic based on the context, version, product selected in the GUI or included in the export. Calculated values are generated when clicking on the object in GUI or when exported and are not stored in the database. The foundation for a calculated attribute is a functional programming language very similar to the language used for defining functions in Excel.

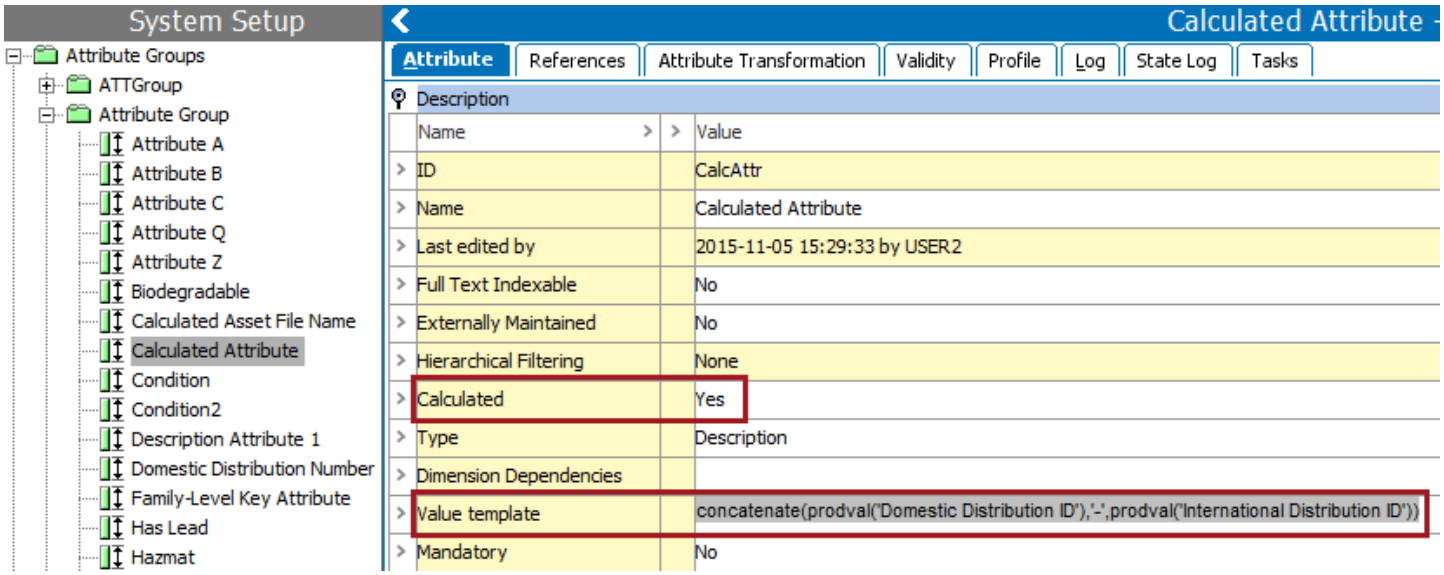
Note: Change Flags for events require that data is present in the database. Since calculated attribute values are not stored, no change flags are generated. For more information, refer to the **Limitations and Exceptions** section of the **Core Events** topic in the **System Setup** documentation.

For assistance in determining if a calculated attribute is the most efficient way to meet your requirements, refer to the **Calculated Attribute Considerations and Limitations** topic in the **Calculated Attributes** section of the **System Setup** documentation.

The value of a calculated attribute is determined by a value template which can be written in the Function Editor. Many functions can include an argument, which specifies the STEP data to be accessed. Functions can access information from references, compare data between objects, round numbers, modify text, and many other options. For a step-by-step guide to creating a calculated attribute, refer to the **Creating a Calculated Attribute** topic in the **Calculated Attributes** section of the **System Setup** documentation.

As an example, the following functions and arguments are included in the workbench image below:

- The 'concatenate' function takes a comma-delimited list of arguments. In this example, the three (3) arguments are: prodval('Domestic Distribution ID'), '-', and prodval('International Distribution ID').
- The 'prodval' function takes an argument of an attribute ID and retrieves the value of the attribute identified. In this example, the two (2) attribute IDs are: 'Domestic Distribution ID' and 'International Distribution ID'.
- Assuming that the 'Domestic Distribution ID' value is 1234 and the 'International Distribution ID' value is 5678, the result of this 'Value template' would be 1234-5678.



For additional use cases for calculated attributes, refer to the **Calculated Attribute Use Cases** topic.

In Web UI, calculated attributes can be identified and their values can be overridden on the Attribute Management screen as described in the **Calculated Attributes in Web UI** section of the **Web User Interfaces** documentation.

Considerations and Limitations

Generally, business actions are preferred to calculated attributes with the understanding that the value is not always up to date, typically immediately following approval.

While calculated attributes provide a lot of flexibility, they also have a direct impact on downstream deliveries. Additionally, calculated attributes can degrade performance as defined below.

- For cross-context exports, STEP attempts optimizations to avoid repetitive calculations per context, but the following limitations exist:

  Attributes used in the calculated attribute must not be dimension dependent. The cross-context exporter does not impact performance significantly when extracting non-calculated attributes in many contexts compared to a single context. However, the time it takes to extract calculated attribute values grows in a nearly linear way with the number of contexts. STEP cannot automatically detect if the formula that expresses the calculation is dimension dependent or not. This means that introducing many calculated attributes for new contexts to a downstream delivery can significantly degrade performance.

  Iterations in a calculation must not include references or child objects. Only attribute values directly on the selected object or inherited values for a product are allowed. Locality refers to the distance between the values required by a calculation and the calculated attribute. Calculated attributes that access only values on the local object can be very efficient, but calculated attributes that navigate to other objects (via references and/or hierarchy links) are more expensive in terms of performance. Therefore, calculated attributes that navigate to other objects should be carried out in a controlled and limited manner.

Calculated attributes must not rely on the value from another calculated attribute. Chained calculated attributes depend on other calculated attributes and can make performance unpredictable.

- Calculated attributes that cross transitive closures can potentially grow very large cause performance to degrade. For example, a calculated attribute that involves all children of a large (and growing) entity or product hierarchy.
- Calculated attributes are calculated each time they are viewed.
- Complex calculated attributes increase the load time of an object (product or entity) in the Web UI and workbench.

Recommendations

Review calculated attributes by exporting all attributes as Excel or CSV and filter on 'Attribute Calculated'. Add the identified calculated attributes to a collection and export the collection as STEPXML to evaluate the value templates for the following elements.

- Identify the products and entities that take a long time to load. In workbench, from the View menu, click **Disable calculated values** and verify if there is a measurable difference between viewing these objects without the calculated attribute values. Significant differences in load time indicate further analysis is needed as outlined in the following items. Refer to the **View Menu** topic in the **Getting Started** documentation.
- Check the log files for SEVERE errors of 'DAPTreeWalker.' This indicates that a calculated attribute value template uses the function 'stepurl' to call a reference instead of a node. Correct the calculated attribute value template to resolve this error. For more information, refer to the **Logs** topic in the **Administration Portal** documentation.
- If the calculated attribute is included in cross-context exports, consider performing the calculation on approval via a business action, and then copy the result to a non-calculated attribute.
- Avoid calculated attributes that navigate to other objects via references and/or hierarchy links across children. Replace the calculates attribute with a business action.
- Avoid calculated attributes across transitive closures that potentially grow very large.
- Avoid having chained calculated attributes and use business actions for better performance.
- Consider In-Memory to optimize the performance of reading data on complex data models via calculated attributes which navigate references. For more information, refer to the **In-Memory Database Component for STEP** section of the **Resource Materials** section of online help.

LOV Filtering Recommendations

This is one of the recommendations for attribute and attribute groups. The full list is defined in the **Attribute and Attribute Group Recommendations** topic.

Filtering a List of Values (LOV) can eliminate duplicate LOV values, while restricting the values available for selection based on the location of an object in the hierarchy, or based on the attribute using the LOV. However, this functionality can result in poor performance and be difficult to manage.

Avoid using LOV filters to enforce data population as the functionality is intended to support filtered end user selection rather than globally restricted data population. Specifically, when LOV filters are implemented, users are prevented from manually selecting values outside of the applicable filter.

For more information, refer to the **Filtering LOVs** topic in the **System Setup** documentation.

The following sections discuss filters for LOVs that are set on the LOV itself or using the LOV attribute and hierarchical filtering.

Filtering on the LOV

The LOV itself can be filtered via the 'Value Filter' on the List Of Values, as defined in the **Filtering LOV Values** topic in the **System Setup** documentation and shown in the table below. This enables you to define legal values based on the attribute using the LOV.

'Color' LOV Values	Legal Values for Attribute 1	Legal Values for Attribute 2
• Green • Red • Yellow • Blue • White	Filter Includes • Green • Yellow • White	Filter Includes • Green • Red • Blue

Filtering on the LOV attribute and hierarchical filtering

The LOV attribute can be filtered via the Hierarchical Filtering parameter, as defined in the **Filtering by Hierarchy on LOV Attributes** topic in the **System Setup** documentation and shown in the table below. This enables you to define legal values based on product and/or classification hierarchies.

'ColorLOV' Values	Legal Values for the 'Color' attribute on the Plastic Products Node	Legal Values for the 'Color' attribute on the Wood Products Node
• Green • Red • Yellow • Oak • Mahogany	Filter Includes • Green • Red • Yellow	Filter Includes • Oak • Mahogany

Note: Hierarchical LOV filtering operates as an intersection. This means that when a product is associated with multiple classifications, only the common values that are shared across all the filters within these classifications will be considered valid for the given product.

Recommendations

Export all attributes and attribute groups as STEPXML and in the output file search for **HierarchicalFiltering="true"** to review areas for improvement.

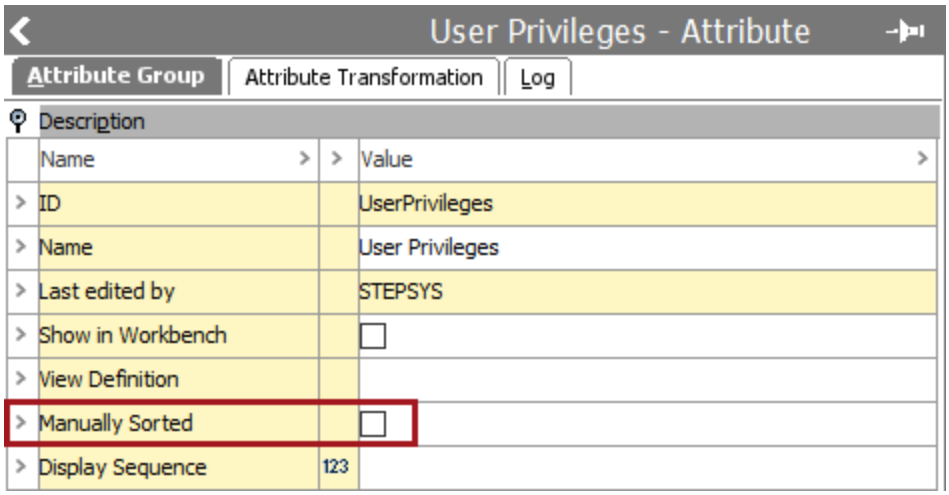
Avoid hierarchical filtering on product and classification hierarchy where possible since hierarchical filtering in both product and classification hierarchies can negatively affect performance.

Avoid using the 'Ignore LOV filters' option on the relevant product-to-classification link types. While it restricts which parts of the classification hierarchies are used for LOV filter inheritance, it can become difficult to manage.

Manually Sorted Recommendations

This is one of the recommendations for attribute and attribute groups. The full list is defined in the **Attribute and Attribute Group Recommendations** topic.

The Manually Sorted option requires additional processing power and can have an impact on performance when using extensively.



Manually Sorted can apply to attribute groups as well as object types.

The limitations are:

- Manual sequence is only set up in the workbench.
- Manual sequence can only be enabled for attribute groups that have no sub-groups.
- Manual sequence is only supported for attributes. Sequencing of 'tag groups,' LOV groups and reference types below an attribute group are not supported.

For more information, refer to the Manually Sorted section of the **Attribute Groups** topic in the **System Setup** documentation.

To review the attribute groups with this option enabled, export all attribute groups as STEPXML and in the output file search for **ManuallySorted="true"**. Determine if the parameter can be disabled.

To review the object types with this option enabled, export all object types as Advanced STEPXML using the following template and in the output file search for **ManuallySorted="true"**. Determine if the parameter can be disabled.

```
<STEP-ProductInformation>
<UserTypes ExportSize="All"/>
<EdgeTypes/>
```

```
<CrossReferenceTypes ExportSize="All"/>
</STEP-ProductInformation>
```

Recommendations

Use the Manually Sorted option only when required to sequence object types or attributes in their attribute group.

Background Process Queue Recommendations

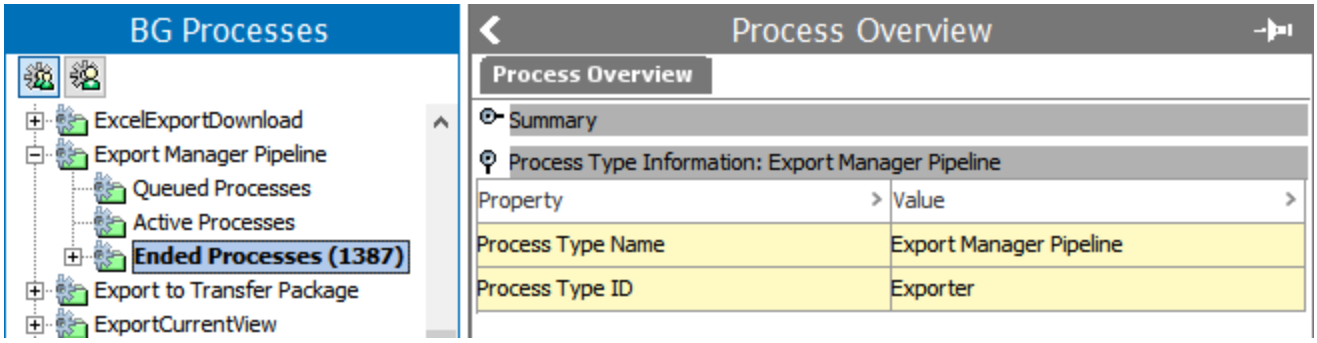
This is one of the data gathering methodologies and recommendations for functional performance improvement. The full list is defined in the **Performance Recommendations** topic.

Configuration properties allow admin users to manage system behavior for a wide range of functionality, for example, the management of background processes and web services.

> **Note:** Ensure changes in the settings are applied with care and properly tested before promoting to production. Parallel setting and multi-threading can easily result in optimistic locking issues, which can have a negative impact on performance, rather than the desired performance gain.

Parallel and multi-threading optimizations can be applied to background process (BGP) queues used by integration endpoints, event processors, and scheduled processes.

To determine the background process type and the queue handling the process, in workbench, on the BG Processes tab, click the all users button ( ), select a node, open the Process Type Information flipper, and find the 'Process Type ID' parameter.



Recommendations

Legacy background process (BGP) functionality uses specified queues, while the recommended BGP execution mechanism runs BGPs based on the priority of the BGP and the created time. Refer to the **BG Processes Execution Management** topic in the **System Setup** documentation.

The following legacy recommendations apply to specific types of BGPs.

- Long-running integration endpoints: use the legacy 'Queue for the endpoint process' parameter, the queue size, the 'Transactional settings' parameter, and CPU cores as appropriate to limit optimistic locking issues.
- Event processors: use the legacy queue size and CPU cores as appropriate to limit optimistic locking issues.

- Scheduled processes (types include BulkUpdate, Exporter, etc., and others like AsyncJobRunner): use the queue size, the 'Transactional settings' parameter, queue parallel, and CPU cores as appropriate to limit optimistic and pessimistic locking issues.

Review the topics below for information on running parallel and multi-threading processes.

In the **System Setup** documentation:

- **Background Processes and Queues** - for explanations and examples of parallel and multi-threading properties
- **Default Configuration for Background Process Queues** - for standard configuration settings
- **Modifying Background Process Queue Configuration** - for instructions on updating queue settings to use parallel and multithreading

In the **System Administration** documentation:

- **Event Processor and Event Queue Recommendations** - for instructions on managing queues and queue size
- **Import Elements to Use** - for information about using separate queues for important IIEPs
- **Export Elements to Use** - for information about multi-threading and using separate queues for important IIEPs

In the **Data Exchange** documentation:

- **Integration Endpoint Transactional Settings** - for information about managing the data processing order

Base Setup Recommendations

This is one of the data gathering methodologies and recommendations for functional performance improvement. The full list is defined in the **Performance Recommendations** topic.

The base setup of the contexts and dimensions, product and entity data model, assets and classifications are based on business requirements. Nevertheless, the base setup should also be designed based on performance since a complex base setup can have a negative impact on general performance.

An analysis of the base setup is worthwhile, as changes on the base setup for performance reasons usually have a great impact.

For example, changing the data model may impact attributes and references, but can also impact business rules, workflows, Web UI configurations, exports, and imports.

Oracle KODO Database Layer

The Oracle KODO layer is the database layer which allows for Java Persistence API (JPA) specification and Java Data Objects (JDO). By default, the Oracle KODO layer has a cache of 10,000 data relations on an object. An excess of 10,000 relations overwhelms the cache and negatively impacts performance. Examples include a hierarchy with more than 10,000 children, a product with more than 10,000 references, a list of values with more than 10,000 values, and an outbound integration endpoint with a batch size of more than 10,000 events.

Storing large data relations can have a significant negative performance impact due to the likelihood of having a large number of the related objects that are no longer in the cache.

If your system performance is significantly impacted due to this limitation, contact Stibo Systems Support to request a configuration setting to increase the data cache size in the sharedconfig.properties file.

Important: Always weigh the pros and cons of changing the base setup prior to making the change, and test the effects on a lower system before making change on a production system.

For detailed information, refer to the following topics:

- Asset Recommendations
- Classification Recommendations
- Data Model Recommendations
- Dimension and Context Recommendations
- Global Count of Object and Attribute Recommendations
- Reference Recommendations

Asset Recommendations

This is one of the data gathering methodologies and recommendations for base setup improvement. The full list is defined in the **Base Setup Recommendations** topic.

> **Note:** These recommendations are valid for systems without In-Memory. The performance costs are different with In-Memory. For more information, refer to the **In-Memory Database Component for STEP** section of the **Resource Materials** in online help.

Digital assets are media files like images, videos, documents, etc. In STEP, these are asset objects which can hold attributes (metadata) and have a reference to the actual asset (binary file).

Storage Location

The asset binaries can be stored outside STEP in an external DAM system. The storage of the asset binary can also be stored on the file system or in the STEP database. While storing the asset binaries in the database might seemingly allow faster access, accessing many asset binaries in the database also results in higher database traffic. Therefore, asset binary storage in the file system or in the STEP database does not determine the performance of STEP.

Dimension Dependency

Dimension dependency on assets should be avoided when it is not required. In workbench, on System Setup, asset dimension dependency is defined in the Users & Groups node under the Image & Document Settings.

Image & Document Settings	
Dimension Dependencies	
Store assets and DTP documents in	Database
DTP asset source	Asset Push
Pregenerate thumbnail cache on upload	Yes
Disable auto-cleanup of thumbnail cache	No
Transformation Lookup Tables follow asset dimension dependency	N
Asset Import Compatibility Mode	Simple

The Dimension Dependencies option is used if there is a requirement to store images or documents in different contexts. For example, you may have a requirement store an image that has English labels in the image and the same image with French labels, as found in the examples below.

To set up, click in the Dimension Dependencies value row, then click the ellipsis button (...). Select the Language option. Now the image in English can be stored in the English context and the image in French can be stored in the French context.

Note: In the image examples used above, both must have the same image ID when loading into STEP and the appropriate dimension point must be selected for each image.

Import and Export

STEP allows for asset transformations during import and export. Asset transformations require system resources and therefore should be avoided unless necessary. For more information, refer to the inbound and outbound sections of the **Digital Assets** documentation.

Asset import optimizations include:

- Using the standard asset importer, as defined in the **Asset Importer** topic of the **Digital Assets** documentation.
- Import assets into a folder hierarchy instead of using a single folder, including saving uncategorized assets into subfolders. The standard asset importer can automatically create an asset folder structure.

Classification Recommendations

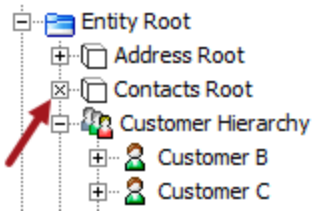
This is one of the data gathering methodologies and recommendations for base setup improvement. The full list is defined in the **Base Setup Recommendations** topic.

> **Note:** These recommendations are valid for systems without In-Memory. The performance costs are different with In-Memory. For more information, refer to the **In-Memory Database Component for STEP** section of the **Resource Materials** in online help.

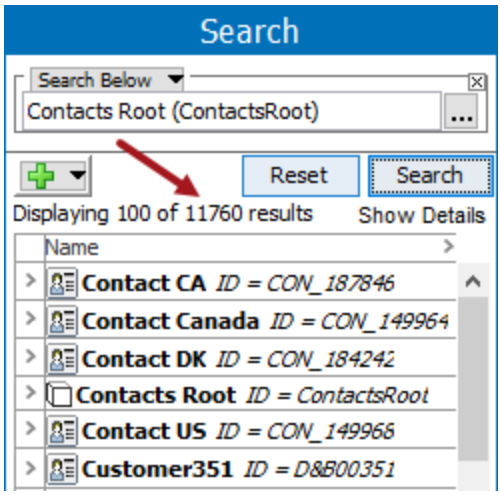
STEP allows for setting up one primary hierarchy (blue folders) and multiple classifications (yellow folders). However, many children in a level of a hierarchy is expensive from a performance perspective. The built-in STEP unique name check means that having many items in one node of the hierarchy can degrade performance.

A classification or hierarchy with more than 10,000 children should be avoided. For details and an alternative, refer to the **Oracle KODO Database Layer** section of the **Base Setup Recommendations** topic.

In Tree, after attempting to expand a hierarchy with 'too many' children, the node is identified with ⊠ (shown in the image below). Nodes that cannot be expanded are automatically set to non-readable and cannot be browsed. 'Too many' for workbench is determined by the case-sensitive property **Workbench.TreeNode.MaxNumberOfChildrenToShowInTree** and for Web UI, the case-sensitive property **Portal.TreeNode.MaxNumberOfChildrenToShowInTree**, both are displayed by accessing the STEP System Administration button on the Start page. For more information, refer to the **Configuration** topic in the **Administration Portal** documentation.



For nodes that cannot be expanded, Search Below can be used to analyze how many children are in the hierarchy. For example, searching below the 'Contacts Root' hierarchy shows it has 11,760 children.



Recommendations

Use the following recommendations to optimize performance:

- Avoid classification hierarchies with 10,000 children by organizing and categorizing children in sub hierarchies which allow you to manage millions of child nodes.
- Instead of a functional classification, use an alphanumeric classification naming structure to accelerate the listing of many objects by creating subcategories at import or by applying auto-classification.
- Identify long-running business rules via the Activity Dashboards tab as defined in the **Admin Portal Business Rule Activity Dashboard** section of the **Business Rule Analysis** topic. In workbench, analyze the business rule to determine if the nodes being modified are excessively large.

Data Model Recommendations

This is one of the data gathering methodologies and recommendations for base setup improvement. The full list is defined in the **Base Setup Recommendations** topic.

Note: These recommendations are valid for systems without In-Memory. The performance costs are different with In-Memory. For more information, refer to the **In-Memory Database Component for STEP** section of the **Resource Materials** in online help.

The data model of STEP is very flexible and allows for complex data model setups. While the data model is designed and set up based on business requirements, the data model should also avoid unnecessary complexity and ensure optimal performance. A complex data model can have a negative impact on the general performance.

Important: For performance reasons, do not over-design the data model since a complex data model requires more extensive processing. Also, use the 'Manually Sorted' parameter on product objects and entities sparingly. For more information, refer to the **Attribute Groups** topic in the **System Setup** documentation.

The following example data models highlight the strengths of various approaches for achieving the business requirements.

Data Containers

Consider a required 'Company' entity that has a visiting and distribution address with country, city, street, and zip code. The addresses can be defined using one of the following methods:

- Attributes directly on the 'Company' entity, such as visiting address street, visiting address city, distribution address zip code, etc.
- A separate 'Address' entity for each address type (e.g., 'visiting address' and 'distribution address'), street, city, zip code, etc., and reference the 'Company' to the 'Address.'
- Two data containers 'Visiting Address' and 'Distribution Address' with country, city, street, zip code, etc.

The preferred data model to use is likely the data containers, since:

- Reusing the same address over multiple companies is minimal. Therefore, the data model with the 'Address' as a separate entity is unnecessarily complex.
- Some companies might not have a 'Distribution Address,' making the usage of separate attributes on 'Company' somewhat inflexible.

Product References

Consider that a product belongs to one product family in most cases, but in some exceptional cases might also belong to another product family. The products can be defined using one of the following methods:

- The product in the blue hierarchy belongs to a parent product family in the blue hierarchy. For the exceptional cases (where the product also belongs to another product family), a product reference is used where the product is also related to the other product family.
- The product-override is used where the product-overrides are alternate versions of products and product families that may have differing values, references, links, and structures. Attributes and values applied to the product family are inherited to the product-override and can be replaced with local values and references on the product-override.

The preferred data model to use is likely the product reference, when inheritance of the second product family is not required, since it is less complex.

Dimension and Context Recommendations

This is one of the data gathering methodologies and recommendations for base setup improvement. The full list is defined in the **Base Setup Recommendations** topic.

Note: These recommendations are valid for systems without In-Memory. The performance costs are different with In-Memory. For more information, refer to the **In-Memory Database Component for STEP** section of the **Resource Materials** in online help.

The setup of contexts and dimensions is flexible and is often designed and configured based on business requirements. Ideally, the setup of contexts and dimensions should be based on avoiding unnecessary complexity and optimal performance as well, since a complex setup can have a negative impact on general performance.

Usually, in multilingual setups, two dimension points are defined by:

- Country
- Language

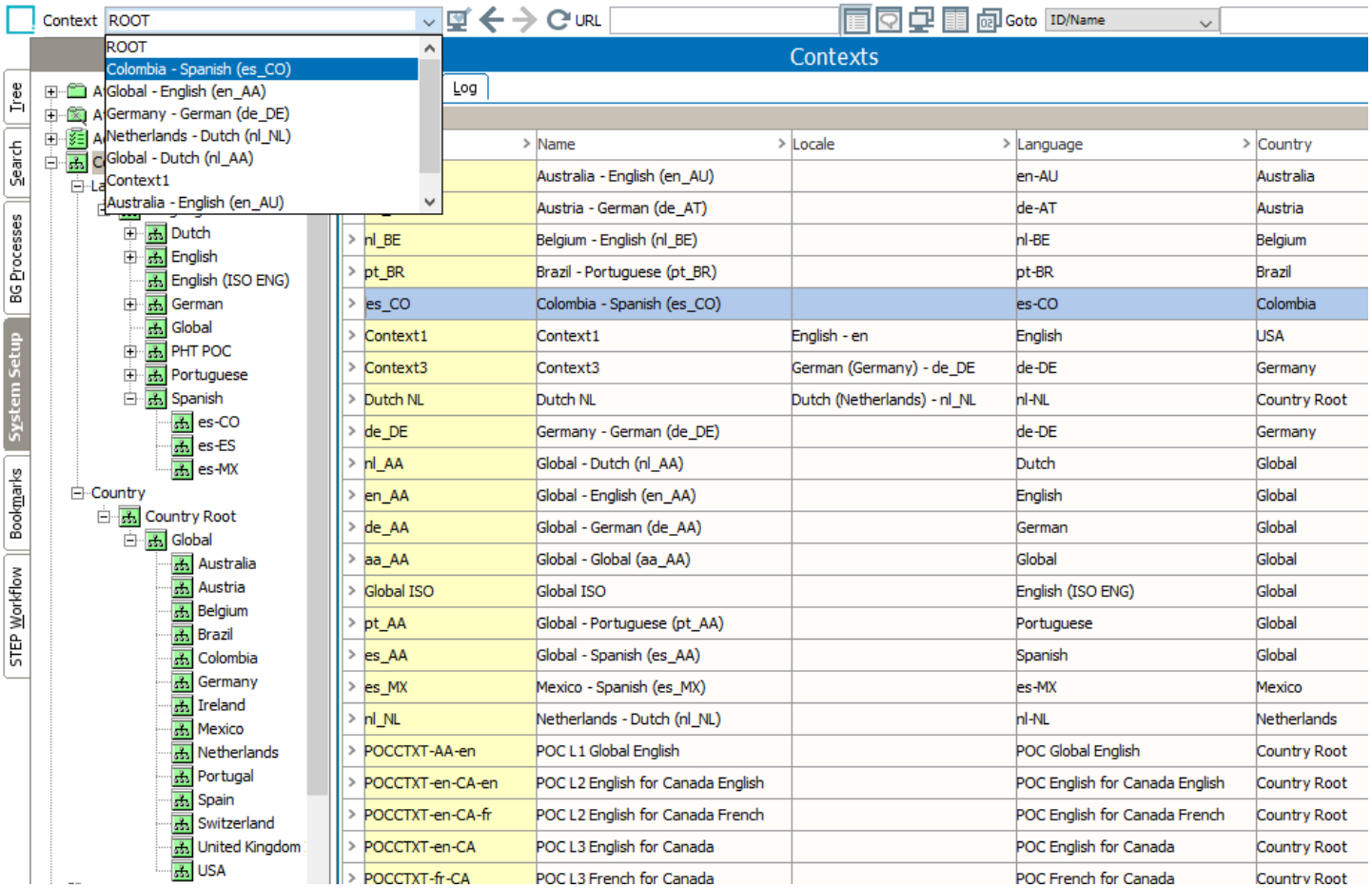
For this example, every object (both entities and products) can have a country and a language dimension:

- The country dimension allows differences in countries, such as different accessories of the product in different countries via suppressing a reference.
- The language dimension allows translations in different languages where the same object has an English, German, Spanish, etc., translation of the data.

The context defines the combination between the defined dimension points.

For example, the context **Colombia - Spanish (es_CO)** is defined with:

- Country = Colombia for Colombia (es-CO)
- Language = Spanish for Columbia (es-CO)



Important: The dimensions allow setup in a hierarchy using inheritance. This means that the dimension point inherits the data from its parent unless it has its own data. For example, the language 'Spanish for Colombia (es-CO)' is setup below the language 'Spanish (es).' This means that all data in the language 'Spanish for Colombia (es-CO)' is inherited from the language 'Spanish (es)' until local data in the language 'Spanish for Colombia (es-CO)' is set.

Once the dimensions and contexts are configured, then attributes, references, etc., can be configured to be dimension dependent.

For example, an attribute 'Consumer Description' can be configured to be language dependent.

Attribute | References | Attribute Transformation | Validity | Profile

Description

Name	Value
ID	Consumer Description
Name	Consumer Description
Last edited by	2018-01-25 10:43:29 by STEPSYS
Full Text Indexable	No
Externally Maintained	No
Completeness Score	
Hierarchical Filtering	None
Calculated	No
Type	Specification
Dimension Dependencies	Language;
Mandatory	Yes

And a reference such as 'Accessory' can be configured to be country dependent.

Reference Type | Validity | Log

Description

Name	Value
ID	SI-Accessory
Name	Accessories for Sales Items
Last edited by	2017-06-02 10:09:24.0 by STEPSYS
Externally Maintained	No
Dimension Dependencies	Country;
Completeness Score	

Recommendations

The following scenarios should be considered when determining the necessary dimension points for the system:

- Setting more than one dimension on an object can result in a complex situation and should be avoided. For instance, if a metadata attribute for an asset is defined to be language and country dependent, then it is possible to have a translated metadata attribute on the asset in different languages, and have a different metadata attribute value for different countries. It is hard to predict values when making an attribute multi-dimensional, such as when the metadata attribute inherits the values from its parents (e.g., from language Spanish and from country Global).
- Only define two dimensions (e.g., language and country) and only configure dimension dependency when required. For example, if the objects do not differ much in the countries, then only define a language dimension.

The more dimensions defined, the more complex the solution will be. A complex dimension setup requires more extensive processing in imports, exports, bulk updates, running business rules, etc.

The number of dimension points (e.g., number of languages) is of less importance, as well as the number of contexts.

For more information, refer to the **Dimensions, Dimension Points, and Contexts** topic in the **System Setup** documentation.

Global Count of Object and Attribute Recommendations

This is one of the data gathering methodologies and recommendations for base setup improvement. The full list is defined in the **Base Setup Recommendations** topic.

> **Note:** These recommendations are valid for systems without In-Memory. The performance costs are different with In-Memory. For more information, refer to the **In-Memory Database Component for STEP** section of the **Resource Materials** in online help.

From the Start page, click the About STEP link, and log in to display counts of a variety of objects.

- System name: **dreloc**
- STEP version: **10.2-mp1-2021-07-08-14-09-06**
- Detailed version information
- Number of user accounts (including suppliers): **63**
- Number of supplier accounts: **7**
- Allowed number of user accounts: **100**
- Number of products (approx.): **900**
- Number of classifications (approx.): **8000**
- Number of assets (approx.): **500**
- Number of entities (approx.): **7000**
- Number of contexts: **12**
- Number of dimensions: **2**
- Is InMemory Enabled: **false**
- Number of languages: **10**

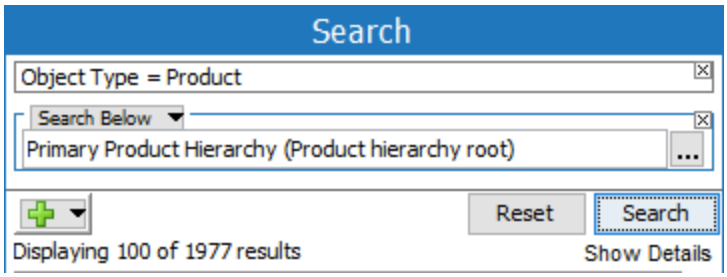
Additionally, the following are examples of using search to count the number of objects in STEP. The search displays the total number of results found.

An example is included for the following types of counts:

- Products
- Products by object type
- Entities
- Entities by object type
- Assets
- Attributes

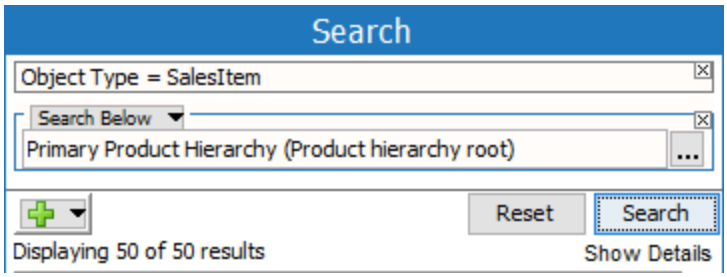
Count the number of products

Search criteria: 'Object Type = Product' and 'Search Below = Primary Product Hierarchy'



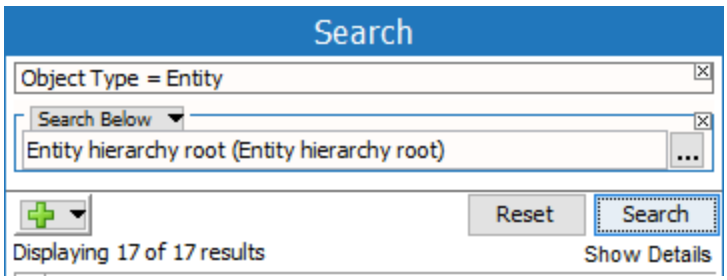
Count the number of products by object type

Search criteria: 'Object Type = [*Product Object Type*]' and 'Search Below = Primary Product Hierarchy'



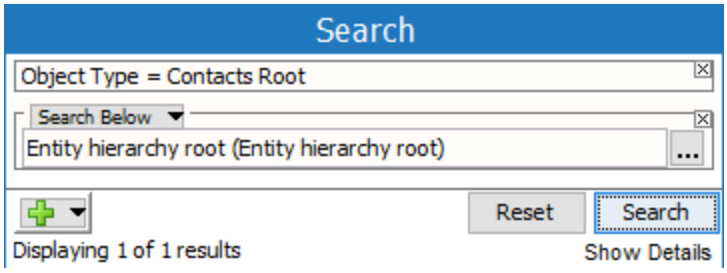
Count the number entities

Search criteria: 'Object Type = Entity' and 'Search Below = Entity Hierarchy Root'



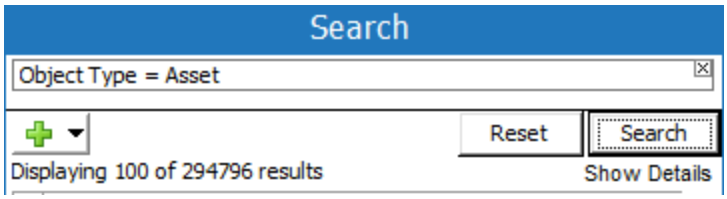
Count the number entities by object type

Search criteria: 'Object Type = [*Entity Object Type*]' and 'Search Below = Entity Hierarchy Root'



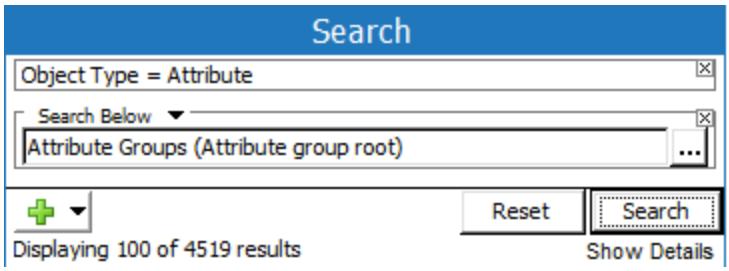
Count the number of assets

Search criteria: 'Object Type = Asset'



Count the number of attributes

Search criteria: 'Object Type = Attribute' and 'Search Below = Attribute Groups'



Reference Recommendations

This is one of the data gathering methodologies and recommendations for base setup improvement. The full list is defined in the **Base Setup Recommendations** topic.

> **Note:** These recommendations are valid for systems without In-Memory. The performance costs are different with In-Memory. For more information, refer to the **In-Memory Database Component for STEP** section of the **Resource Materials** in online help.

A few of the flexible elements of references include:

- Ability to configure with multiple different sources and targets (reference and reference by)
- Are of a certain type (product, asset, classification, etc.)
- Can have metadata attributes

None of these reference configurations independently have a direct impact on performance.

However, objects with many references can have a negative impact on performance since the object needs to be re-approved every time a reference changes. Also, since the approval of an object checks each reference, when the object has thousands of references, approval can become costly in performance. For details, refer to the **Oracle KODO Database Layer** section of the **Base Setup Recommendations** topic.

Inheritance also has an impact on performance because the hierarchy is navigated to identify valid values.

Additionally, navigation across a reference to another object, to fetch attribute values and/or references, is expensive on performance. This is because another database access, and potentially another disk access, is required. For example, navigation via references in business rules to resolve values, resolving calculated attributes, or resolving references during export.

Recommendations

When encountering either of the cases above:

- Verify that references are owned by the object type that requires the fewest number of approvals. An object that owns many references carries the load of the approval process, including verifying approval for all referenced objects.
- Limit the usage of references in business rules, calculated attributes and during export.

Business Rule Recommendations

This is one of the data gathering methodologies and recommendations for functional performance improvement. The full list is defined in the **Performance Recommendations** topic.

Business rules allow extension of the STEP functionality with customer-specific business logic. Business rules provide a flexible way to tailor the core functionality in a very precise manner and can be tied to, e.g., bulk updates, events, imports and IIEPs, approval processes, workflows, Web UI screens, etc.

Important: The system load of the business rule execution can have effect on the performance of STEP. Running multiple complex business rules degrades performance while STEP is processing the business rules.

Database lock issues are common when very long business rules perform changes while running in parallel. To avoid locking issues on a busy, multi-user system, a business rule must run inside a single transaction and the transactions must be short to allow for several commits during the overall processing. For more information, refer to the **Avoid Large Transactions** section in the **Business Rule Elements to Avoid** topic.

The available types of business rules, and the purposes for which each is optimized, are defined in the **Business Rules** topic in the **Business Rules** documentation.

Details about where business rules can be used are defined in the **Using Business Rules in STEP** topic in the **Business Rules** documentation.

Optimizing performance in business rules involves the following:

- Business Rule Analysis
- Business Rule Elements to Use
- Business Rule Elements to Avoid

Extending STEP Functionality

If the business rule recommended practices have been implemented, and additional performance improvement is required, consider the following additional ways to extend STEP functionality.

Scripting API	Allows customers and partners to expand STEP functionality with JavaScript Business Rules. Involves a low level of customization and extended functionality.
Extension API	Allows customers and partners to extend STEP functionality with

	development of plug-ins and components. Involves a moderate level of customization and extended functionality.
Custom Extensions	Allows Stibo Systems to extend STEP functionality with development of custom extensions. Involves a high level of customization and extended functionality.

Using a Java extension can sometimes improve performance over a JavaScript business rule.

For example, this provides the ability to develop a custom extension instead of executing many complex business rules on import of data.

Note: Only cases where extreme amounts of logic are executed should be considered for this solution. In such cases, consider why it is necessary to run such complex logic.

The possible benefits of using a custom extension include:

- Event batching is used by the processor to ensure that business logic is executed exactly once per product per batch, even if multiple imports were executed for a single product.
- Optionally, the example plugin uses the parallel processing framework to maximize performance. This can potentially allow a 'strict' transaction endpoint to still benefit from parallel processing in that rule execution can be multi-threaded while still ensuring data integrity.
- Resolve optimistic locks and deadlocks in cases where the locks are caused not by the import itself, but by logic that accesses and writes to objects shared among parallel import processes.

  A common example of this occurs with tree structures. If parallel imports are executing on children of a common parent, each of which executes business logic causing an update to the parent, the result is deadlocks and sever performance degradation.

  Another performance-related issue with this pattern is that the business logic is executed once per child. However, if the import logic is changed to republish the parent to the event queue instead of each child, event batching will result in a single update to the parent object, regardless of the number of children imported.

- The same concept can be applied using an OIEP configured with a business rule pre-processor. Execute import business logic in the pre-processor, then discard the event so that no exports are produced by the endpoint. Since endpoints do not use multi-threading while executing the pre-processor, the execution time may be longer than if a custom extension is used.

This solution also has some consequences:

- There will be a delay between importing data onto the object, and the business rule running. If timing is an issue, consider other options.
- The object cannot be inspected in both its previous and current form as easily, the business rules running in such a processor will have less information available than the same rule running during import.

Business Rule Analysis

This is one of the recommendations for performing analysis on business rule to improve performance. The full list is defined in the **Business Rule Analysis** topic.

Business rules are units of business logic that are stored as objects in System Setup. Business rules are used for many different purposes in STEP and come in three variants:

	Input	Output	Side effects allowed
Business actions	Current object, current event batch, etc. provided by the context in which the action is executed. For more on business actions, refer to the **Business Actions** topic.	None	Yes
Business conditions	Current object, current event, etc. provided by the context in which the condition is evaluated. For more on business conditions, refer to the **Business Conditions** topic.	Boolean result of evaluating the condition and a message for the user	No
Business functions	Input parameters defined by the function and provided by the functionality evaluating the function. For more on business functions, refer to the **Business Functions** topic.	Result of evaluating the function	No

A fourth type of business rule, **business library**, allows users to define JavaScript library functions that can be called from other JavaScript-based business rules. For more information on business libraries, refer to **Business Libraries** topic documentation.

For more information on differentiating between scenarios where one business rule is more useful than another, refer to the **Business Rule Use Cases** topic in the **Business Rules** documentation.

Using Business Rules in STEP

Wherever business rules are used, they are always tested or executed in relation to one object at a time, and in a specific context / workspace. The most common places to use business rules are included in the list below.

- **Approvals** - Conditions can be tested when approval is attempted on an object under revision control, and the condition can allow or prevent the approval. Actions executed on approval and can modify data in STEP (typically data on the object being approved), send emails etc. related to the approval.

  For more information, refer to **Business Rules on Approval** documentation.

- **Automatic Classifications** - Actions can be executed to automatically classify objects, and can be applied, for example, on approval, during an import, or as a part of a workflow.

  For more information, refer to **Using Automatic Classification with Business Actions** documentation.

- **Bulk Updates** - Conditions can be tested as a precondition for executing an action. Actions can be executed.

  For more information, refer to **Run Business Rule Operation** documentation.

- **Conditional Attributes** - The JavaScript business action 'Conditionally Invalid Values' bind resolves to a set of all values.

  For more information, refer to **Business Rules with Conditional Attributes** documentation.

- **Data Profiles** - Conditions can be tested against all objects in a category (for example, a part of the Product hierarchy), and the result of the tests can be displayed on the Profile Dashboard.

  For more information, refer to **Business Conditions in Data Profiling** documentation.

- **Event Processors** - Actions can determine when and how to act upon the events from the event processor. This is useful when changes occur on objects and custom actions need to occur.

  For more information, refer to **Execute Business Action Processing Plugin Parameters and Triggers** documentation or **Execute Business Action for Event Batch Processing Plugin Parameters and Triggers** documentation.

- **Gateway Integration Endpoints** - Accessed from JavaScript in business rule conditions and actions, the bind can work with a variety of the REST methods.

  For more information, refer to the **Gateway Integration Endpoint Bind** documentation in the **Resource Materials** online help. *Gateway Integration Endpoints (GIEPs) do not always use the REST plugin; the Gateway Integration Endpoint Bind topic only applies for GIEPs that use the REST plugin.*

- **Imports and Inbound Integration Endpoints** - Conditions can be tested during imports, and the condition can allow or prevent the creation or update of objects. Actions executed during import can modify the objects being imported, apply actions to objects being imported, send emails, and start workflows, etc.

  For more information, refer to **Business Rules in an Import Configuration** documentation.

- **Matching, Linking, and Merging** - Actions and Conditions can be used to normalize the data for comparison to identify the duplicate products in STEP. Actions can also be used in relation to Golden Records Survivorship Rules.

  For more information, refer to Matching section of the **JavaScript Binds** topic in the online help **Resource Materials** documentation or the **Golden Records Survivorship Rules** topic in the **Matching, Linking, and Merging** documentation.

- **Outbound Integration Endpoint** - Conditions can be used as event filters for an event-based OIEP. Actions can be executed via a pre-processor for any OIEP, or as a event generator to generate derived events to export or publish for an event-based OIEP.

  For more information, refer to **OIEP - Event-Based - Event Triggering Definitions Tab** documentation or **OIEP - Pre-processor - Business Action** documentation.

- **Web UI** - Actions can be executed to update the object. Conditions can be evaluated to improve data validity.

  For more information, refer to **Business Rules in Web UI** documentation.

- **Workflows** - Conditions can be tested within workflows to allow or prevent transitions from one state to another. Actions can be executed when entering a state, when leaving a state, when performing a specific transition, and when a deadline is met. Actions can also modify the object being tracked by the workflow, modify other objects in STEP, modify the workflow behavior, send email, start other workflows, etc.

  For more information, refer to **Business Rules in Workflows** documentation.

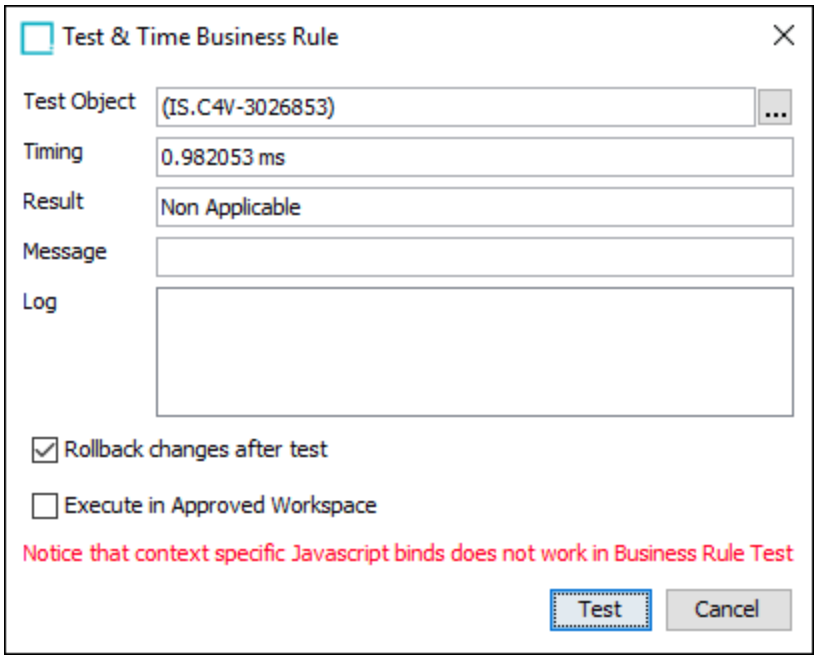
Test & Time Business Rule

Use the 'Test & Time Business Rule' dialog, as defined in the **Testing a Business Rule** topic of the **Business Rules** documentation. This option gives a first indication of the performance of the business rule for a certain item.

Test a business rule multiple times against objects that will either fail or pass and analyze the performance timing.

Test a long-running business rule to verify the performance timing.

For example, the business rule shown below took about 0.98 milliseconds to complete for the selected object. The same business rule may take more or less time for other objects.

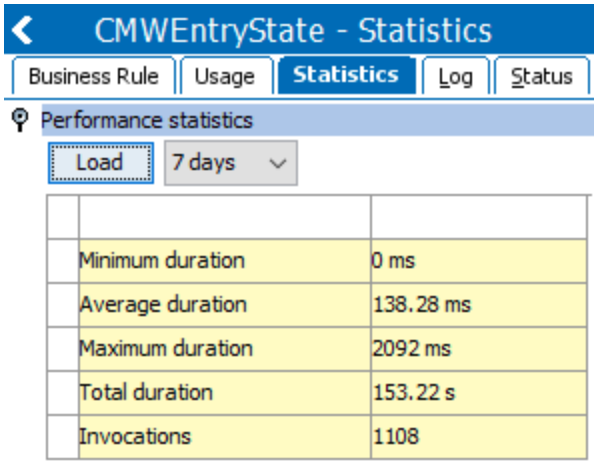


Business Rules Statistics

Use the business rule 'Statistics' tab, as defined in the **Maintaining a Global Business Rule** topic of the **Business Rules** documentation. This tab shows the performance of the business rule over time.

The minimum, maximum, average, and total duration of the business rule, as well as the number of invocations per selected period are displayed. The period can be configured to be between an hour up to a week.

For example, the image below shows the same business rule was invoked more than 100 times during the last 7 days. That average duration was about 138 ms. Click on the maximum duration of about 2092 milliseconds to show which item the business rule took longest to execute.

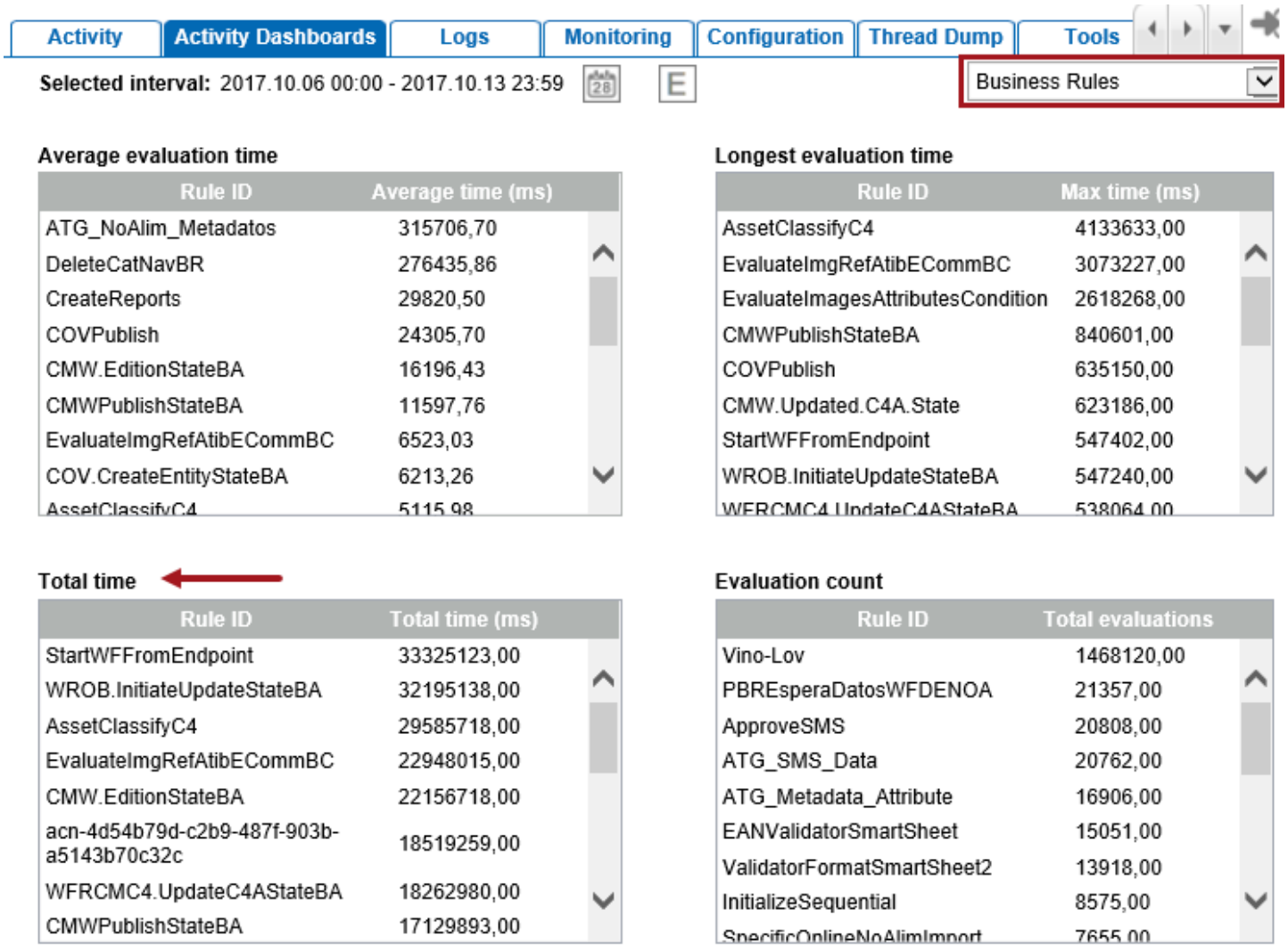


Admin Portal Business Rule Activity Dashboard

Use the Admin Portal 'Activity Dashboards' tab to display business rules information, as defined in the **Activity Dashboards** topic of the **Administration Portal** documentation. This option allows you to track and trace the most demanding business rules performance over a given period.

The dashboard shows the top business rules over the configured period, with the longest average evaluation time, the longest maximum evaluation time, the longest total time, and the number of invocations.

Start with the 'Total time' section to review the business rules with the longest average evaluation time and the most number of invocations.



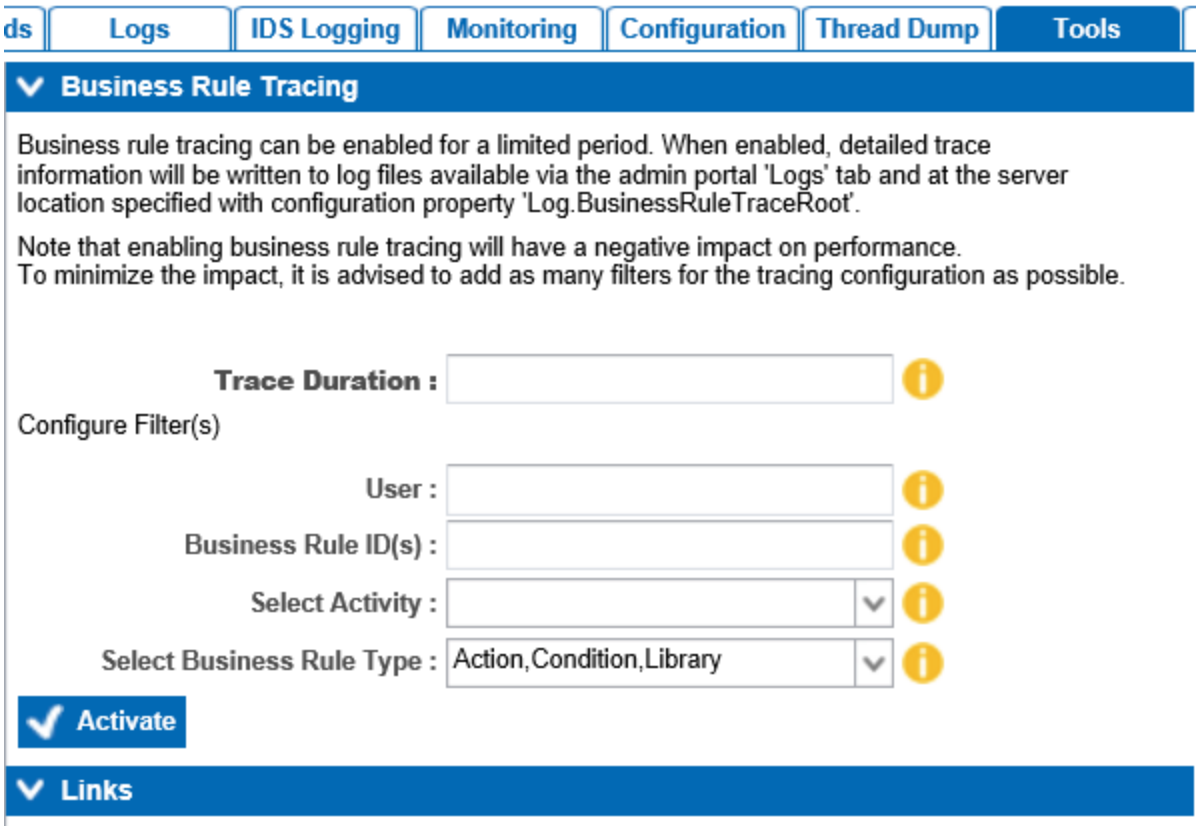
Admin Portal Business Rule Tracing

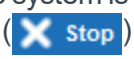
Use the Admin Portal 'Tools' tab to run the Business Rule Tracing functionality. When enabled, business rule tracing writes to log files.

> **Important:** Enabling business rule tracing has a negative impact on performance. To minimize the impact, leave it enabled for a limited time and add as many tracing configuration filters as possible.

Define the location of the tracing log files by setting the case-sensitive property **Log.BusinessRuleTraceRoot** in the sharedconfig.properties file.

To enable business rule tracing, click the yellow information icon next to each parameter for a description of the parameter / filter. Supply the relevant information and click the **Activate** button. Then within the 'Trace Duration' time frame, from workbench, trigger the business rule(s) being traced.



Tracing stops when the system is stopped or restarted, and when the Trace Duration expires. Once a trace is activated a stop button (

) is displayed, which allows a user to manually stop tracing.

Business Rule Elements to Use

This is one of the recommendations for performing analysis on business rule to improve performance. The full list is defined in the **Business Rule Analysis** topic.

The following list includes business rule elements known to assist system performance. This list can be used to troubleshoot existing long-running business rules, and can also be reviewed prior to writing new business rules to prevent performance problems.

Use Exception Handling

If an error occurs during approval, an exception is thrown from the domain layer. If this exception is caught in a business rule but not re-thrown, it will not reach the exception approval handler. In this case, objects can be inconsistently approved (some parts are approved and other parts are not). This behavior also has a negative effect on the performance of the business rule.

When writing JavaScript business rules, it is important that 'try…catch' statements are designed correctly. 'Try...catch' statements should not swallow (catch and ignore) exceptions that should cause changes made by the script to be rolled back and cause the business rule to fail.

Additionally, only catching an exception without rethrowing it can lead to inconsistent data or the error: 'The transaction cannot be committed, because it was already marked for rollback only. The transaction will be rolled back instead.'

Important: Carelessly catching and ignoring exceptions will lead to derived errors, which usually makes it difficult to determine the root cause.

The sample code in the table below demonstrates a way to log exceptions in JavaScript without swallowing those that should be handled by the framework.

Correct	Incorrect

`try {`
`  // Some code`
`} catch (e) {`
`  logger.info(e);`
`  throw(e); // REQUIRED`
`}`
	`try {`
	`  // Some code`
	`} catch(e) {`
	`  logger.info(e);`
	
	`}`

The only types of JavaScript exceptions that may be caught and handled locally are:

- Checked exceptions thrown by an invoked method (including any subclasses of the specified checked exception) (refer to the example below)
- Exceptions not generated by calls to the STEP API

Important: Any other exception must always be re-thrown if caught, or not caught at all.

Correct Handling of Checked Exceptions

The code snippet below demonstrates how a checked exception can be caught and handled locally, including the necessary Rhino specific '**e.javaException**' notation.

```
try {
currentObject.createReference(targetAsset, primaryImageRefType.getID());
} catch (e) {
if (e.javaException instanceof com.stibo.core.domain.UniqueConstraintException) {
logger.info("Reference could not be created");
} else {
throw(e); // All other exceptions MUST be re-thrown
}
}
```

Proper exception handling correctly re-throws the exception when using 'try-catch' in business rules and avoids inconsistent objects.

For more information, refer to the **Recommended Error Handling Practices** topic of the **Resource Materials** section of online help.

Use Logging Carefully

Warnings and errors encountered while executing business rules can be written in the Main STEP Log File. While logging many details can have a negative impact on performance, using the correct logging level can aid in troubleshooting and resolving unexpected outcomes.

Logging can be managed for all business rules or on a per-business rule basis.

Manage Logging for all Business Rules

Define the appropriate logging level for each STEP server environment using the following case-sensitive entries in the sharedconfig.properties file:

- **BusinessRule.Warning.Threshold** - a threshold in milliseconds for business action execution. If it takes longer to execute or test a given business action, a warning is posted in the Main STEP Log File.

For example, the following setting writes a warning to the STEP log file whenever a business rule execution takes longer than 10 seconds.

```
BusinessRule.Warning.Threshold=10000
```

- **Log.Level** - specifies the level of detail for logging business rules results, the entry shown below would include log entries for info, warning, and severe:

```
Log.Level = INFO
```

- **Log.Level.org.mozilla.javascript.MemberBox** - specify the logging level for business rule results, as shown below:

```
Log.Level.org.mozilla.javascript.MemberBox = SEVERE
```

The logging values are ALL, FINEST, FINER, FINE, CONFIG, INFO, WARNING, SEVERE, and OFF. The most granular logging provided using ALL, while minimal logging is written using SEVERE. Setting a level also includes logging for the levels to the right. INFO is the default setting.

Suggested settings are FINE to trace errors on a development or test system, INFO or WARNING for a QA system, and SEVERE on a production system.

For more details about the log levels, refer to the **Server Log File Settings** topic in this guide.

Manage Logging per Business Rule

Log the result of business rules during development on the development server, but turn off the logging when deploying to the test, quality, and production servers. An easy and transparent way to control logging per business rule is to set a debug flag in the business rule code, as illustrated below.

```
//Debug 'flag' DO NOT use unless you develop or test
//When doing tests DO NOT test on large amount of products
//REMEMBER to set to 'false' when development and testing phases are complete
var isDebug = false;

//Function to handle whatever logging of debug information should occur or not
function logDebug(message) {
if(isDebug) {logger.info(message)}
}
...
logDebug("A message for the log file")
...
```

Use Arrays, Not Multiple Read Calls

Business rules repeatedly using calls to the database for large sets of data significantly degrades performance. Instead, use one call to get the data, and push it into arrays and work from there. Minimizing the number of calls to the database aids performance.

When multiple business rules are executed sequentially (e.g., as part of an approval process), and these business rules fetch the same data from the database multiple times, it is beneficial to rewrite the business rules to fetch the data once, and push the data into (multi-dimensional) arrays.

In-Memory

In-Memory can improve performance of the business rules because it provides faster operations on complex data models where business rules navigate references.

In-Memory may improve performance on business rules that still perform poorly after implementing the recommended practices for business rules. For more information, refer to the **In-Memory Database Component for STEP** topic of the **Resource Materials** section of online help.

Consider Using an Extension

An extension should be considered when additional performance improvement on business logic in the system is required and all previous recommendations on business rules are implemented (including In-Memory). Some of the code using in JavaScript (business rules) might run faster in Java (extensions).

Important: Only cases where extreme amounts of logic are executed should be considered for this solution. And in such cases, first consider why it is necessary to run such complex logic.

For example, you could develop a custom extension instead of executing many complex business rules on import of data. The possible benefits of such a solution are:

- Event batching is used by the processor to ensure that business logic is executed exactly once per product per batch, even if multiple or many imports are executed for a single product.
- The example plugin optionally uses the parallel processing framework to maximize performance. Potentially, this allows a 'strict' transaction endpoint to benefit from parallelization in that rule execution can be multi-threaded while still ensuring data integrity.
- Optimistic locks and deadlocks can be resolved in cases where the locks are caused not by the import itself, but by logic that accesses and writes to objects shared among parallel import processes.

  A common example of this occurs with tree structures. If parallel imports are executing on children of a common parent, each of which executes business logic causing an update to the parent, deadlocks (and sever performance degradation) will result.

  Another performance-related issue with this pattern is that the business logic is executed once per child. However, if the import logic is changed to republish the parent to the event queue instead of each child, event batching will result is a single update to the parent object, regardless of the number of children imported.

- For STEP 7.4+ solutions, the same concept can be applied using an outbound integration endpoint configured with a business rule pre-processor. First, execute import business logic in the pre-processor, then discard the event so that no exports are produced by the endpoint.

Note: Endpoints do not use multi-threading while executing the pre-processor so the execution time may be longer than if a custom extension is used.

However, a pre-processor solution also has some consequences:

- There will be a delay between importing data onto the object and the business rule running. If timing is an issue, consider another option.
- Running business rules in a pre-processor does not allow the object to be inspected in both its previous and current form as easily as running the same rule running during import.

Business Rule Elements to Avoid

This is one of the recommendations for performing analysis on business rule to improve performance. The full list is defined in the **Business Rule Analysis** topic.

The following list includes business rule elements known to degrade system performance. This list can be used to troubleshoot existing long-running business rules, and can also be reviewed prior to writing new business rules to prevent performance problems.

Avoid Large Transactions

Business actions involve a transaction which allows you to manipulate data in STEP. Business actions with long-running transactions degrade the performance. Additionally, since STEP runs with optimistic locking policy, large transactions increase the probability of optimistic locking failures when running the business action simultaneously. For more information, refer to the **Optimistic and Pessimistic Locking Recommendations** topic.

Keeping business rule transactions small means it is necessary to develop business rules while considering the worst-case scenario. This includes the following recommendations:

- Avoid traversing a substantial percentage of the complete data range.
- Keep changes to data local (to the nodes nearest your main data object).
- Avoid business rules that follow and potentially change the objects of transitive closures of referential structures, such as everything underneath a high-level folder in the product hierarchy containing thousands of children. Instead, use a bulk update or a customer specific background process.

If it seems that the business rule transactions cannot follow these recommendations, refer to the **'Reference Target Lock Policy' Parameter** section of the **Optimistic and Pessimistic Locking Recommendations** topic.

Avoid Large Libraries

On each execution, JavaScript libraries are compiled. Each dependency is stacked into a script which is also compiled using the ScriptEngine.eval() method before each execution of the business rule.

For example, Script S depends on Library A and Library B. Library A depends on B. In this case, the script is stacked as follows:

- A::script
- A::B::script
- B::script
- S::bindings
- S::script

If a business rule depends on a library, the library is compiled as well. By extension, if a business rule uses only a single function within a library, the whole library is still compiled.

And if a library depends on another library, the other library needs to be compiled as well.

STEP caches the scripts instead of reloading them from the database. By default, 100 business rules are cached. Generally, it takes about 500 milliseconds to compile about 8,500 lines of code at each business rule execution.

Important: Be aware that libraries are compiled every time the business rule is executed, which is especially burdensome to performance when libraries depend on each other, and are used in many business rules. Dividing a large library into multiple libraries, but keeping the dependencies, does not resolve the issue.

To improve performance, consider making the library functions local to the business rule.

Avoid Many Inclusions of a Library

Avoid smaller libraries with functions used multiple times by the business rule. For example, a business rule that is applied to 10,000 objects can take up to five times as long to complete compared to running the same function locally in the business rule, instead of from a library.

Evaluate poor performing business rules to determine if one of these scenarios applies:

- If one or more function(s) are called from a **large** library (thousands of lines), make the library functions local to the business rule itself or use a business function and analyze the level of performance improvement.
- If one or more function(s) are called from a **small** library but the function is used many times (thousands of executions), make the library functions local to the business rule itself or move the business rule itself to the library or use a business function and analyze the performance improvement.

Avoid Infinite Loops

Infinite loops lead to severe degradation of system performance on the affected application server(s). Ultimately, an infinite loop can make the entire STEP installation unresponsive.

Analyze business rules to determine if infinite loops exist.

Avoid the 'getChildren' Function with Many Nodes

Business rules that use 'getChildren' on a large number of children (more than 10,000) can cause memory problems because the function reads all the children. Instead, use 'queryChildren' function, which limits the number of children affected.

Analyze business rules to determine if 'getChildren' is used on a selection with more than 10,000 children, and update the 'queryChildren' function as needed.

Avoid Updating Data via Business Conditions

Business conditions are optimized for determining the true / false result of read-only scenarios. For details, refer to the **Business Conditions** topic in the **Business Rules** documentation.

Event Processor and Event Queue Recommendations

This is one of the data gathering methodologies and recommendations for functional performance improvement. The full list is defined in the **Performance Recommendations** topic.

Event processors and queues are used for asynchronous processing, such as auto-purging revisions on schedule or performing a delayed approval after import. Warnings and/or critical issues in asynchronous processing can have an influence on the performance of the system.

For general information on event processors and event queues, refer to the **Event Processors** section and **Event Queues** section, both in the **System Setup** documentation.

For more information, refer to the **Monitoring** topic of the **Administration Portal** documentation.

Run Without Warnings and Errors

Warnings and errors are visible in the event processor background processes. For optimal processing, no warnings or errors should be reported.

For example, the following image shows the business rule execution report for an event processor. In this illustration, the event processor initiated a business rule with a setting 'Valid Object Types = All object types valid' but the event processor is not valid for all object types. Business rules initiated via an event processor should be defined with specific object type(s) rather than setting all object types as valid.

```
27 Processed batch with 1 events. (Mon Nov 18 12:27:05 CET 2019)
28 Business Action Add_LeadingZerosToGTIN is not applicable for event: Node: Init, Object Type: Produkt
29 Business Action Add_LeadingZerosToGTIN is not applicable for event: Node: Init, Object Type: Produkt
```

Optimize STEP Setup for Performance

Use the following setup recommendations within the event processor configuration to optimize performance. For detail on all parameters, refer to the **Event Processors** topic in the **System Setup** documentation.

- The **User Running Event Processor Plugin** parameter defines the privileges that are checked when the event processor is invoked. For each piece of information processed, all privileges are checked for the selected user. Creating a user with relatively few and broad permissions explicitly for event processors can improve performance dramatically.
- The **Number of Events to Batch** parameter defines the number of events processed in a batch. While the events are handled one-by-one, committing large batches are more effective and improve performance. For the event processors:

- Set the 'Number of events to batch' parameter no larger than the max size defined in the sharedconfig.properties file for the OutboundMessageProcessor.BatchEventsMaxSize property. By default, this is 10,000. Lower this number if you experience system memory problems.
- Analyze the schedules and the background processes to determine how many events are processed in each batch.

For example, in the following image, the number of events in a batch is 1 or 2, so the number of events to batch could be set to five (5). However, if the schedule is increased, e.g., once per 15 minutes instead of once every minute, then the number of events in a batch could be an average of 30. In that case, the 'Number of events to batch' parameter could be set to 30.

```
2  Processed batch with 2 events. (Fri Nov 15 09:12:32 CET 2019)
3  Processed batch with 1 events. (Fri Nov 15 09:36:39 CET 2019)
4  Processed batch with 2 events. (Fri Nov 15 10:35:59 CET 2019)
5  Processed batch with 1 events. (Fri Nov 15 10:45:00 CET 2019)
6  Processed batch with 2 events. (Fri Nov 15 10:52:04 CET 2019)
7  Processed batch with 2 events. (Fri Nov 15 11:05:09 CET 2019)
8  Processed batch with 2 events. (Fri Nov 15 11:07:10 CET 2019)
9  Processed batch with 1 events. (Fri Nov 15 11:43:18 CET 2019)
10 Processed batch with 1 events. (Fri Nov 15 11:47:19 CET 2019)
11 Processed batch with 1 events. (Fri Nov 15 12:10:25 CET 2019)
12 Processed batch with 1 events. (Mon Nov 18 08:03:22 CET 2019)
13 Processed batch with 1 events. (Mon Nov 18 08:14:28 CET 2019)
14 Processed batch with 1 events. (Mon Nov 18 08:42:56 CET 2019)
15 Processed batch with 1 events. (Mon Nov 18 08:46:58 CET 2019)
16 Processed batch with 1 events. (Mon Nov 18 09:06:08 CET 2019)
17 Processed batch with 2 events. (Mon Nov 18 09:22:15 CET 2019)
18 Processed batch with 2 events. (Mon Nov 18 09:29:20 CET 2019)
```

- The **Days to Retain Events** parameter is the number of days to keep events once processed. However, the value should be set to 0 since this parameter has no impact on event processors.
- The **Schedule** parameter can be set to run every minute, but this will have a negative impact on performance. The 'Start every minute' option should be used only in rare and necessary situations.

Analyze Asynchronous Processing

Analyze the state of asynchronous processing using the 'Sensors for external monitoring' option as follows.

1. On the Start page, click the **STEP System Administration** button, and supply the login credentials.
2. On the **Monitoring** tab, open the Additional Links section and click the 'Sensors for external monitoring' link to display the list of monitors.
3. For each sensor with a 'Warning' or 'Critical' status, click the sensor name link to display the details of the status.
4. Resolve the issue indicated in the sensor log.

For example, the following sensor has a 'Critical' status.

Sensor	Status	Message
EventQueueSensor-DynamoDBEvents	**Critical**	1649000 events have been queued, which is more than the scheduled limit of100000

The details show that event queue 'DynamoDBEvents' has too many events queued.

Sensor status for EventQueueSensor-DynamoDBEvents

Plugin	EventQueueSensor
Sensor	DynamoDBEvents
Status	**Critical**
Created	Wed Oct 31 12:49:15 UTC 2018 (0 seconds ago)
TTL	30 seconds
Short message	1810000 events have been queued, which is more than the scheduled limit of100000

Performance data

Name	Value	Unit	Warning	Critical	Min	Max
Estimated number of unread events	1810000.0					

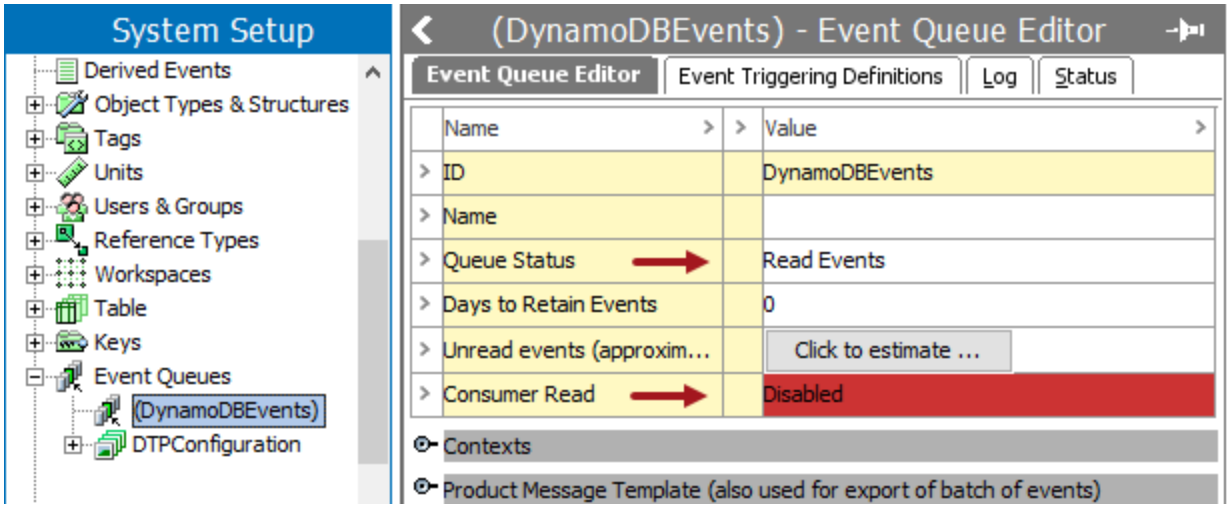
Formats

The status shown on this page is also available in the following machine-friendly formats:

- A simple status string, Possible values: OK, WARNING, CRITICAL, UNKNOWN.
- Nagios plugin output, output formatted for easy integration with Nagios.
- Full xml all available data in xml for easy parsing by ad-hoc monitoring tools.

Please do not rely on the output of this page for automated monitoring, use one of the formats above.

In the workbench, this event queue shows that the event queue is disabled but is still reading events. Since the latest change on this event queue is months ago, and it is on the production environment, the event queue is no longer being used and can be removed.



Legacy - Queues and Queue Size

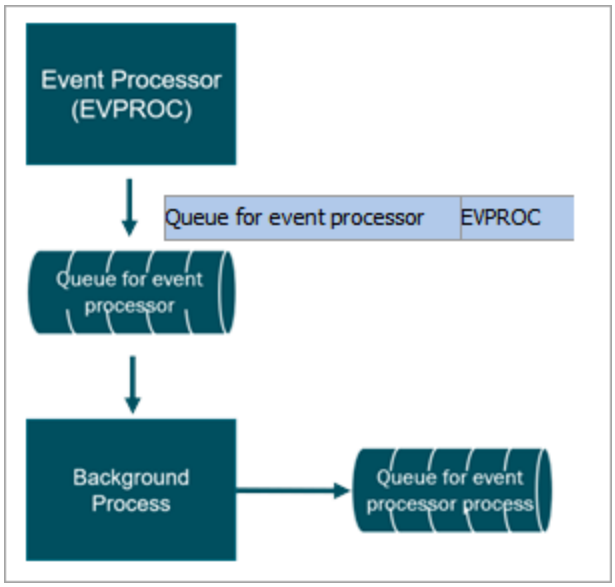
Legacy background process (BGP) functionality uses specified queues, while the recommended BGP execution mechanism runs BGPs based on the priority of the BGP and the created time. Refer to the **BG Processes Execution Management** topic in the **System Setup** documentation.

The event processor initiates a background process to handle the processing.

- The legacy **Queue for event processor** parameter stores the queue that is used by the background process to poll the event processor. The default value is **EVPROC**.

- The queue for associated processes is used by the background processes to handle the actual processing. The queue is identified on the BG Processes tab: select the Event Processor BGP node to display the Process Overview tab and open the 'Process Type Information' flipper. The default value is **EventProcBGP**. For details on identifying and editing the queue being used, refer to the **Modifying Background Process Queue Configuration** topic in the **System Setup** documentation.

STEP legacy functionality allows you to define a separate queue for event processors to ensure only events for this process are read.



The first time you activate the event processor, a queue with the specified name is created if it does not already exist.

A background process (BGP) queue allows prioritizing system processes to ensure high- level system performance. BGP queues are named and configured to control which processes run on which server and how many can run at the same time on each server.

Note: Parallel processing is set by increasing the queue size for the event processor background processes and can improve performance, but can also cause optimistic locking issues when parallel processes attempt to view / update the same node at the same moment. A queue size of 1 means only one process can run at a time, while a queue size of 2 means two processes can run concurrently.

Recommendations

Configure the simplified BGP execution mechanism to run BGPs based on the priority of the BGP and the created time. Refer to the **BG Processes Execution Management** topic in the **System Setup** documentation.

In the event processor settings, legacy functionality uses separate queues for event processors, especially for those with long-running events.

Test parallel background processing using an increased the queue size in a test environment before increasing it in production. Increase the queue size to 2 for the event processor background process via the following case-sensitive properties in the sharedconfig.properties file:

- BackgroundProcess.ProcessType.[*process type ID*].Queue=[*queue*]

  For example:

  ```
  BackgroundProcess.ProcessType.EventProcBGP.Queue=EventProcBGPQ
  ```

- BackgroundProcess.Queue.[*name*].Size=[*number of allowed concurrent processes*]

  For example:

  ```
  BackgroundProcess.Queue.EventProcBGPQ.Size=2
  ```

- Restart the application server to apply the changes to the properties file.

Refer to the **Background Processes and Queues** topic in the **System Setup** documentation for examples of parallel and multi-threading properties.

Consider In-Memory for Event Processors

When the other event processor recommendations have been followed and additional performance improvement is needed, consider In-Memory since it provides faster read operations.

For more information, refer to the **In-Memory Database Component for STEP** topic in the **Resource Materials** section of online help.

Export Recommendations

This is one of the data gathering methodologies and recommendations for functional performance improvement. The full list is defined in the **Performance Recommendations** topic.

One of the primary goals of most STEP implementations is to reduce the time to market required by the flow of data from STEP to external systems such as ERP systems and e-commerce systems.

Important: Optimizing the export performance may adversely affect the performance of the system as a whole. For example, multithreading can increase export performance, although it may have a negative impact on the overall system performance.

The available tools for exporting data are defined in the **Data Exchange** documentation.

Optimizing performance in exports involves the following:

- Export Elements to Use
- Export Elements to Limit

Export Elements to Use

The following list includes export elements known to minimize impact on system performance. This list can be used to troubleshoot existing exports, and can also be reviewed prior to creating new exports to prevent performance problems.

To ensure maximum export performance, the usual strategy is to limit the amount of information exported, or limit the number of times the same node is exported with the same data.

For general information on exports, refer to the **Data Exchange** documentation.

Optimize Object Type Triggering Definitions

Ensure that only required objects are exported using the Triggering Object Types flipper on the outbound integration endpoint (OIEP) Triggering Definitions tab. For example, when exporting to an e-commerce system, only the product itself is potentially relevant. Triggering on other object types risks spending time exporting objects which would be irrelevant downstream.

For more information, refer to the **OIEP - Event-Based - Event Triggering Definitions Tab** topic of the **Data Exchange** documentation.

Optimize Attribute and Reference Triggering Definitions

Applying triggering definitions can ensure that objects are only exported when necessary (based on the attributes changed) and minimize performance issues. On the outbound integration endpoint (OIEP) Event Triggering Definitions tab, set the object types in the Triggering Object Types flipper. For example, an e-commerce system is probably only interested when certain attributes, like the description, of a product object changes. So the Triggering Object Type is set to 'product' and the Triggering Attributes is set to 'description.'

Additionally, set the OIEP triggering definitions specifically for the export recipient. Create attribute groups specifically for export channels, and then trigger attributes specified by attribute group. For example, the attribute 'WebsiteAttribute' contains only the attribute relevant for the website export channel.

For more information, refer to the **OIEP - Event-Based - Event Triggering Definitions Tab** topic of the **Data Exchange** documentation.

Optimize Event Filter and Event Generator Triggering Definitions

Event filters and event generators are business rules that are executed during export.

For example, consider that inheritance affects changes to attributes residing higher up the hierarchy, and can result in a significant number of products if all child products are exported. In this case, replacing a single high-level attribute change event with multiple child-level events is recommended. Using the event filter / event generator options means the significant volume of events can be handled in multiple batches, which enables output files to be kept to a reasonable size. It also allows exporting multiple batches in parallel reducing overall export times.

Typically, the 'approval' mechanism should not determine if data is suitable for publishing since approval is much earlier within the product lifecycle. Often, it is desirable to limit the data exported to just 'published' objects. The event filter and pre-processor functionality can meet both of these requirements, although they operate in slightly different ways. Alternatively, all data can be exported downstream (related to the underlying events) and a middleware-based solution can determine which data to route to which target system.

For more information, refer to the **OIEP - Event-Based - Event Triggering Definitions Tab** topic of the **Data Exchange** documentation.

Use Multiple Dedicated OIEPs

In many cases, a single OIEP is used to monitor all product changes, i.e., both attribute and reference changes.

In this case, it is easy to understand how the assignment of many products to a classification, e.g., 'Christmas Gifts' can cause a significant volume of data to be exported, delivered, and processed by the receiving system, even though no actual product data has changed.

This issue can be avoided through the use of multiple OIEPs:

- One OIEP listens for traditional product (attribute) changes and outputs all data for the product.
- Another OIEP listens for reference changes and outputs minimal data, e.g., product ID and target ID.

Note: Generally speaking, the performance gain can be degraded if using more than 10 OIEPs.

For more information, refer to the **Creating an Event-Based Outbound Integration Endpoint** topic of the **Data Exchange** documentation.

Use Multithreading

Increasing the number of threads for a given OIEP can increase export performance for event-based OIEPs.

Multithreading is effective when a large amount of data goes to a downstream system on a regular basis and the downstream system can handle the load.

Consider the following points before increasing the number of threads to more than one:

• STEP system hardware requires enough resources to perform multithreading.

• The downstream (receiving) system must be able to handle parallel events.

Since the batch-fetching of the events runs serially, increasing the thread number results in each batch size being divided by the thread number so that the contents of a batch can be processed in parallel. The data is distributed as evenly as possible to each thread. From fetching the batch through the delivery stage, the pre-processing, main processing, and post-processing take place in parallel. When all threads are complete, the batch yields a single message as is consistent with the 'Strict' transactional setting required by event-based OIEPs. This allows events to be delivered as if everything was executed serially.

Thread size is typically increased for critical information where the speed with which the message is produced is key. However, multithreading can have a negative impact on the overall system performance when a large amount of data is involved.

The recommendation is to consider using multithreading to improve production of messages to a downstream system on a regular basis to improve export performance. Run the settings in a test environment, starting with a small amount of data and then increasing it, before implementing it in the production system.

For more information, refer to the **Event-Based OIEP Multithreading Support** topic of the **Data Exchange** documentation.

Note: Setting the 'Maximum Number of Threads' parameter on the Configuration tab has no impact on a Select Object OIEP (which uses a static set of data, not events).

Optimize the Batch Size

The appropriate batch size is typically based on the size of messages and downstream system processing capabilities:

- Use a smaller batch size for larger messages.
- Use a larger batch size for smaller messages.

For individual OIEPs, the batch size is set using the 'Number of events to batch' parameter on the Event Queue Configuration flipper. For more information, refer to the **OIEP - Event-Based - Event Queue Configuration Flipper** topic of the **Data Exchange** documentation.

Note: Use the system-wide case-sensitive **OutboundExportService.BatchEventsMaxSize** property, set in the sharedconfig.properties file, to limit the number of events included within any batch. When the batch size in the workbench is greater than this setting, this setting overrides the workbench to limit data exported in a single export invocation.

Small Batch Size Considerations

A small batch-size, or no batching at all, invokes the export engine and surrounding framework many times (once per batch), incurring a significant overhead. A small batch-size can also cause the same product to be exported multiple times in cases where a product has been approved multiple times in quick succession, or there have been changes to externally maintained data (in this case one event will be created for each external change).

Note: While setting the batch size = 1 may appear to deliver the required results, in practice, there are problems associated with this approach as defined above. Generally, small batch sizes should be avoided.

To publish one STEPXML file per item (product or entity) while using a batch size larger than 1, use the 'Generic XML Splitter' or the 'STEPXML Splitter' post-processor. For more information, refer to the **Configure the Pre-processor and Post-processor** section of the **OIEP - Select Objects - Output Templates Flipper** topic or the **OIEP - Event-Based - Output Templates Flipper** topic in the **Data Exchange** documentation.

Large Batch Size Considerations

A large batch-size (e.g., 1,000 - 10,000 events) invokes the export engine and associated framework fewer times incurring less overhead and reduces the amount of duplicate product data exported.

The database Oracle KODO cache default limit is 10,000. For example, a hierarchy with more than 10,000 children, a product with more than 10,000 references, a list of value with more than 10,000 values, an OIEP with a batch size of more than 10,000 events, etc. Storing very large data relations can have great negative performance impacts, since there is a high likelihood that a large number of the related objects are no longer in the cache.

Recommendations

- Avoid a batch-size of 1 and if required, use a post-processor to publish separate XML files per item.
- Avoid a batch-size of more than 10,000. If there is a reason for a larger batch size, contact Stibo Systems Support to discuss raising the **Install.DataCache.MaxRelationSize** setting to allow for a larger cache size.

Use Cross-Context Exports

One way to limit how much data is being exported, is to export cross-context STEPXML on an endpoint rather than having multiple endpoints, each exporting in their own context. This is because having multiple endpoints also means exporting the non-context sensitive data multiple times.

Do not configure multiple OIEPs each outputting product details for a single context. Instead, for significantly faster overall export time, use a single OIEP configured with multiple contexts. Subsequently, the context splitter post-processor can be used if separate context output files are required.

For more information, refer to the 'Contexts' parameter section on the **OIEP - Configuration Flipper** topic of the **Data Exchange** documentation.

Use Event-Based Exports Over Static Exports

While static exports can be used, it is usually much more efficient to use event messaging via event-based exports.

In reality, for a larger STEP system with several million products, an event-based integration is the only viable approach. Not only from a STEP perspective (the time taken from the STEP side to export all data), but also from the receiving system perspective (the time to import all data).

It is recommended to use event-based exports over selection-based exports whenever possible.

For more information, refer to the **Outbound Integration Endpoints** topic in the **Data Exchange** documentation.

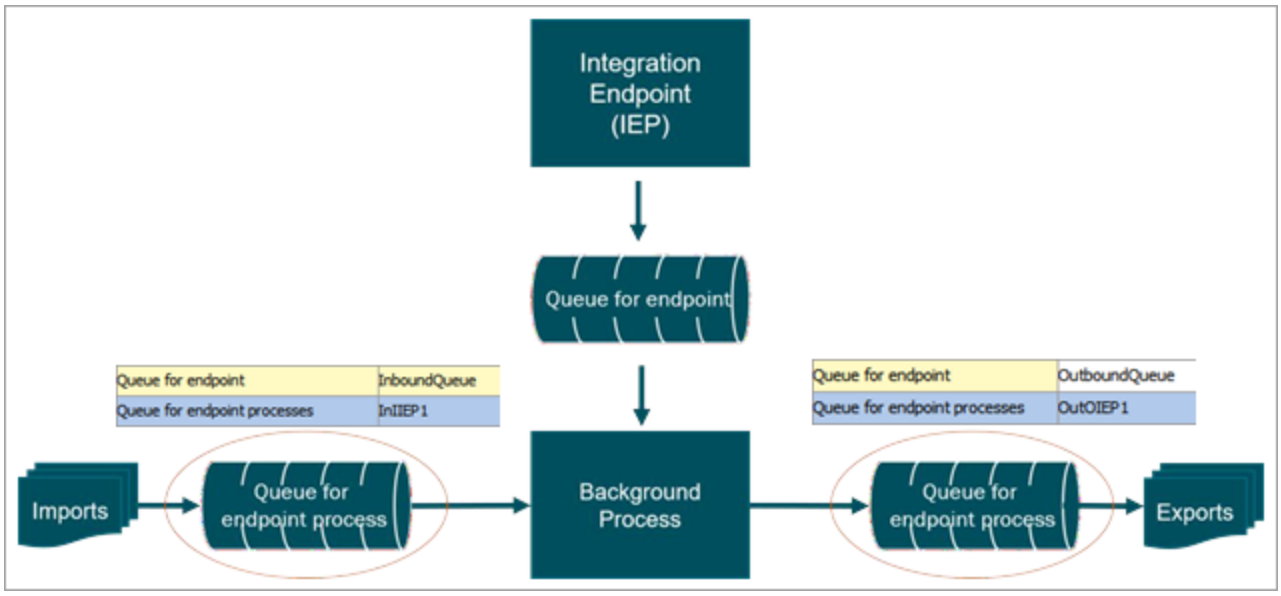
Legacy - Use Separate Queues for Important OIEPs

Legacy background process (BGP) functionality uses specified queues, while the recommended BGP execution mechanism runs BGPs based on the priority of the BGP and the created time. Refer to the **BG Processes Execution Management** topic in the **System Setup** documentation.

The legacy outbound integration endpoint (OIEP) initiates a background process which handles the actual export.

- The legacy **Queue for endpoint** parameter stores the queue that is used by the background process to poll the outbound integration endpoint. The default value is **OutboundQueue**.
- The legacy **Queue for endpoint processes** parameter stores the queue that is used by the background processes to handle the actual export. The default value is **Out**.

STEP allows you to define separate queues for the endpoint and the endpoint process of an OIEP in the Configuration flipper of the OIEP editor. In this example, the queue for endpoint processes for the outbound integration endpoint is renamed to OutOIEP1.



The first time you activate the endpoint, a queue with the specified name is created if it does not already exist. Events are not lost if a separate queue for endpoint process is defined.

Changing the legacy **Queue for the endpoint process** for OIEPs means each OIEP background process uses this named queue for the actual export, causing OIEPs to run simultaneously and not wait for other OIEPs to finish processing. Do not use this when OIEPs require sequential export processing.

Recommendations

Configure the simplified BGP execution mechanism to run BGPs based on the priority of the BGP and the created time. Refer to the **BG Processes Execution Management** topic in the **System Setup** documentation.

Configure separate **Queue for endpoint processes** (to handle the actual export) for all high-priority or long-running OIEPs that do not require sequential processing (exports).

Refer to the **Background Processes and Queues** topic in the **System Setup** documentation for examples of parallel and multi-threading properties.

Consider In-Memory for Exports

In-Memory can improve export performance because In-Memory provides faster read operations. This is beneficial for complex data models, such as export configurations with data aggregations and export configurations with data that navigates references.

For more information, refer to the **In-Memory Database Component for STEP** topic in the **Resource Materials** section of online help.

Optimize STEP Setup for Export Performance

Use the following setup recommendations to optimize export performance.

- Evaluate the export schedules of all exports and schedule long running exports sequentially (one after the other) to balance the STEP system load.
- Remove inactive and unused OIEPs (which includes queues, background processes, and export files, etc.) on operational production environments. When an OIEP is removed, the corresponding background processes are also removed.
- Set OIEPs to limit the number of background processes kept after export using the 'Maximum number of old processes' and 'Maximum age of old processes in hours' parameters, as defined in the **IEP Auto Delete Settings** section of the **Create a Background Processes Maintenance Plan** topic.

Export Elements to Limit

The following list includes export elements known to degrade system performance. This list can be used to troubleshoot existing exports, and can also be reviewed prior to creating new exports to prevent performance problems.

To ensure maximum export performance, the usual strategy is to limit the amount of information exported, or limit the number of times the same node is exported with the same data.

For general information on exports, refer to the **Data Exchange** documentation.

Limit Event-Based OIEPs

Event-based outbound integration endpoints (OIEPs) can adversely affect the performance of the system as a whole. The number of OIEPs can adversely affect the approval / external attribute change process because each change must be checked against the triggering definitions on all OIEPs to determine if any is interested in that specific change. A general guide is to have no more than 10 event-based OIEPs.

For more information, refer to the **Outbound Integration Endpoints** topic in the **Data Exchange** documentation.

Limit the Volume of Exported Data

Advanced STEPXML makes it possible to define what data to export while leaving out the other data. For example, only include the export attribute groups instead of including all attributes. The CSV export format and the Generic XML format allow configuring only the data that is needed.

In the following image, the Advanced STEPXML exports only the 'Accessory' reference type:

```
<ProductCrossReference IncludeInheritedReferences="*" Type="Accessory">
        <Entity Referenced="true">
                <Name/>
                <Values IncludeInherited="true">
                        <Value Derived="true" AttributeID="Training"/>
                </Values>
        </Entity>
</ProductCrossReference>
```

For more information, refer to the **Advanced STEPXML Format** topic and the **Data Formats** topic, both in the **Data Exchange** documentation.

Limit Unnecessary Data

Consider the impact of additional objects and determine if they are required, as many additional objects slows the export process. Generally, limit unnecessary data in the export using these methods:

- An event generator can be used to add additional objects via derived events, as defined in the **Generate Event** section of the **OIEP - Event-Based - Event Triggering Definitions Tab** topic within the **Data Exchange** documentation.
- A pre-processor can add additional objects to the export set associated with the event batch, as defined in the **Configure the Pre-processor and Post-processor** section of the **OIEP - Event-Based - Output Templates Flipper** topic within the **Data Exchange** documentation.
- The Advanced STEPXML template can include additional objects, for example, via references. For more information, refer to the **Filter References in STEPXML** topic within the **Data Exchange** documentation.

Limit Multiple Output Templates

Multiple output templates can be associated with a single outbound integration endpoint (OIEP) enabling the OIEP to handle different types of objects / events, each having the ability to be output in a different format. This flexibility allows the user to change how data is exported for different kinds of objects.

Important: While STEP allows multiple output templates per OIEP, it can have a severe impact on performance since a new batch is created each time a new output template is required.

Alternate approaches include:

- When possible, use different OIEPs to handle the 'family' and 'item' changes separately. This is not an option if the events need to be processed in sequence since the 'item' change could be exported, and delivered, before the 'family' change.
- Use STEPXML to easily contain the 'family' and 'item' data in the same file.

For more information, refer to the **OIEP - Event-Based - Output Templates Flipper** topic within the **Data Exchange** documentation.

Avoid Complex Export Privileges

When exporting, for each piece of information exported, all privileges are checked for the export user. This privilege check often includes a hierarchy check (e.g. 'is the product below a certain root node') which can take a significant amount of time for large exports. Limiting the number of privileges for the exporting user can improve performance dramatically.

Consider creating a user with relatively few and broad permissions explicitly for exports. Since only the user configured on the endpoint is relevant, this export user generally ensures improved performance.

Import Recommendations

This is one of the data gathering methodologies and recommendations for functional performance improvement. The full list is defined in the **Performance Recommendations** topic.

Bulk data imports can be executed manually or automatically on a schedule via an inbound integration endpoint (IIEP). Bulk data imports can vary from fast to slow, depending on the expectations and on following the recommendations for imports described in this section.

A clean, simple, optimized import that does no processing can process about a hundred records per second, whereas a complex import with complex processing may only process one record per second, or perhaps be even slower.

A slow import could be problematic based on the situation. Slow import performance of about one record per second, for example, is fine when a small number of records are imported on a daily basis. However, if the import contains a large number of records and it will take weeks to finalize, then this is likely a problem. This illustrates why it is important to clearly describe the use cases for imports.

If the STEP system suffers from perceived bad performance during import, these recommendations should be considered to optimize performance.

The available tools for importing data are defined in the **Data Exchange** documentation.

Optimizing performance in imports involves the following:

- Import Elements to Use
- Import Elements to Avoid
- Importing for Migration

Import Elements to Use

The following list includes import elements known to minimize impact on system performance. This list can be used to troubleshoot existing imports and can also be reviewed prior to creating new imports to prevent performance problems.

For general information on imports, refer to the **Data Exchange** documentation.

Use Term Lists for Price Data

The structure of the import file can be optimized for maximum import performance by using commercial data (also called terms lists) for price data where appropriate. Especially for complex, time-limited / quantity limited price data, terms lists can speed up imports.

For more information, refer to topics in the **Commercial Data** section of the **Publisher (Adobe InDesign Integration)** documentation.

Use Business Rules Designed for Import Performance

If business rules are required on import, carefully map the business rule execution of each import (especially endpoints) to understand the full impact of the configurations. Ensure any business rules running on import (via approval, import actions, or through a workflow), has acceptable performance.

Review the following items to ensure business rules are used efficiently:

- Simplify complex business rule JavaScript logic being executed on the import.
- When possible, move business actions running on the import to event processors. This only applies if the actions can be performed asynchronously from import.
- Business rules that read or update objects other than the one being imported (reference sources or targets, parents, children, etc.) will reduce performance because only the imported product is likely to be resident in cache at the time. Business rules that iterate through children of the imported product are particularly common, but very expensive performance-wise.
- Consider if it is possible to ensure that no business conditions exist, allowing a logical exception or a lock contention. Business rule execution for the purpose of automation or transformation is secondary and should be treated as fail-tolerant. This means that the criticality of a rule failure is far less than that of the import operation itself. In the event of an exception, rules can be fixed and executed again, whereas import records that are skipped can be difficult to rectify on a busy system.

For more information, refer to the **Business Rule Recommendations** topic.

Use Workflow Initiations Designed for Import Performance

When a product is initiated into a workflow, or a state transition is triggered by an import, all business rules configured on exit of an existing state, transition between, or entry to the next state execute as part of the import process.

The following scenarios can heavily impact the import performance:

- If workflow initiations and/or transitions are necessary, consider if the business rules trigger on entry or exit. The import performance may be heavily impacted by the executing of these transitions.
- If business rules initiate more workflows or auto submit to other states, a single workflow submission may cascade into hundreds (or even thousands) of lines of executed business rule logic.
- When a workflow business rule conditions fails, the current transaction is rolled back.

Use Approvals Designed for Import Performance

If approvals are necessary at import, then consider which approval conditions and actions will be executed.

Ensure that endpoints are importing externally-maintained data, since it requires no approvals. In this scenario, consideration for approval conditions and actions is not required. Also, externally-maintained data has no revision history. Therefore, revision history growth is not a consideration.

Use Event-Based Exports Designed for Import Performance

When importing externally maintained data, or importing and approving, all changes must be checked against any event-based outbound integration end points (OIEPs), to verify if an event should be generated.

Consider which approval events are queued on OIEPs and which event filter / generator rules will be triggered. If a large number of OIEPs exists, and if these have Event Triggering Definitions on attribute groups, for each check, the system must check if the attribute exists under the given attribute group. This can lead to performance degradation.

Ensure that the OIEPs are triggered to as specific and few attributes as possible. For more information, refer to the **Creating an Event-Based Outbound Integration Endpoint** topic in the **Data Exchange** documentation.

Legacy - Use Separate Queues for Important IIEPs

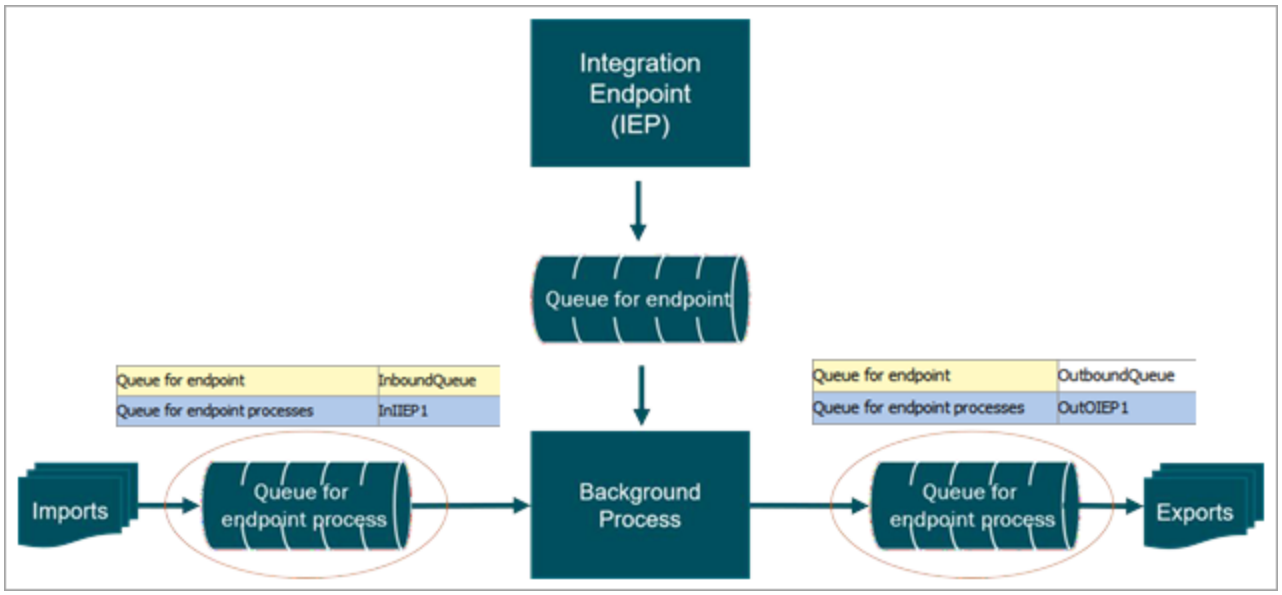
Legacy background process (BGP) functionality uses specified queues, while the recommended BGP execution mechanism runs BGPs based on the priority of the BGP and the created time. Refer to the **BG Processes Execution Management** topic in the **System Setup** documentation.

The inbound integration endpoint (IIEP) initiates a background process which handles the actual import.

- The legacy **Queue for endpoint** parameter stores the queue that is used by the background process to poll the inbound integration endpoint. The default value is **InboundQueue**.

- The legacy **Queue for endpoint processes** parameter stores the queue that is used by the background processes to handle the actual import. The default value is **In**.

STEP allows you to define separate queues for the endpoint and the endpoint process of an IIEP in the Configuration flipper of the IIEP editor. In this example, the queue for endpoint processes for the inbound integration endpoint is renamed to InIIEP1.



The first time you activate the endpoint, a queue with the specified name is created if it does not already exist. Events are not lost if a separate queue for endpoint process is defined.

When changing the **Queue for the endpoint process** for IIEPs, each IIEP background process uses this named queue for the actual import. This means that the IIEPs run simultaneously and not wait for other OIEPs to finish processing. Do not use this when IIEPs require sequential export processing.

Recommendations

Configure separate **Queue for endpoint processes** (to handle the actual import) for all high-priority or long-running IIEPs that do not require sequential processing (imports).

Refer to the **Background Processes and Queues** topic in the **System Setup** documentation for examples of parallel and multi-threading properties.

Optimize STEP Setup for Import Performance

Use the following setup recommendations to optimize import performance.

- Remove inactive and unused IIEPs (which includes queues, background processes, and import files, etc.) on operational production environments. When an IIEP is removed, the corresponding background processes are also removed.
- Set hotfolder and REST Receiver IIEPs to remove files after import using the 'Keep file after load' parameter, as defined in the **Hotfolder Receiver** topic of the **Data Exchange** documentation.
- Set IIEPs to limit the number of background processes kept after import using the 'Maximum number of old processes' and 'Maximum age of old processes in hours' parameters, as defined in the **IEP Auto Delete Settings** section of the **Create a Background Processes Maintenance Plan** topic.
- Use the standard asset importer, instead of the legacy asset importer is used, as defined in the **Importing Assets** topic within the **Digital Assets** documentation.

- Assets and other objects are not imported into a single folder, as defined in the **Hierarchy Builder** topic within the **Digital Assets** documentation.
- Evaluate the import schedules of all imports and schedule long running imports sequentially (one after the other) to balance the STEP system load.
- Review high-priority integrations and integrations with long-running processes. The recommended simplified background process (BGP) execution mechanism prioritizes the order in which background processes are handled. Refer to the **BG Processes Execution Management** topic in the **System Setup** documentation.

  In a legacy background process (BGP) implementation, these integrations should have their own queue for endpoint processes where the background process handles the actual import. The queue is automatically created on the system if it does not already exist. For more information, refer to the **IIEP - Configure Endpoint** topic within the **Data Exchange** documentation.

- Consider In-Memory to improve import performance, as defined in the **In-Memory Database Component for STEP** topic in the **Resource Materials** section of online help.
- Limit the number of OIEPs listening on modify events to avoid excessive checking of event validity, as defined in the **Events** topic of the **System Setup** documentation.
- Use the 'Relaxed' locking policy for commonly-referenced products to improve performance of imports, as defined above.
- Use parallel imports only as needed, considering the performance degradation to the rest of the system, as defined above.

Import Elements to Avoid

The following list includes import elements known to degrade system performance. This list can be used to troubleshoot existing imports and can also be reviewed prior to creating new imports to prevent performance problems.

For general information on imports, refer to the **Data Exchange** documentation.

Avoid Typical Import Errors

The following errors are typical for imports:

- Invalid attribute values - These errors can be caused by the attribute setup parameter in the import not matching the parameter in the system. For example, the attribute's validation type, mask, minimum and maximum values, maximum length, LOV, the object's own object type, and so on.
- Object type validity - Even if you load attribute values for an object, not all values will be automatically available in STEP. Attributes must be made valid for an object before the values can be accessed via references. STEP will always load attribute values if the object's type is valid for the attribute, and the attribute values meet the validity criteria. But, the attribute itself must be a valid attribute for that object, that is, linked somewhere in the object hierarchy or classification hierarchy where the object resides.
- Inaccurate STEPXML statements - Performance degrades when the STEPXML import file has the wrong case for the property 'UserTypeID' (such as 'UserTypeId'). If the STEPXML import file has no Parent IDs for the products, then new products are not imported. To skip importing new products, add the reject new product tag in the file header (RejectNewProducts="true") to avoid the missing Parent ID error.

Background Process (BGP) progress and the number of errors and warnings are shown on the Background Processes tab under the Execution Report flipper as follows:

- For manual imports using Import Manager - on the BG Processes tab under the 'Import Manager Pipeline' node
- For IIEPs - on the Background Processes tab click the BGP link to display information for this particular background process. In both cases, on the Background Processes tab, open the Execution Report flipper to are also displayed as shown below.

For a list of common errors, refer to the **Import Error Messages** topic and the **Import Error Message Examples** topic in the **Data Exchange** documentation.

Avoid Missing Reference Targets

Ensure that import files do not have reference nodes that are missing in the file (missing reference targets).

If the reference target does not exist in the import file, then the target does not exist at the time of import. The importer skips the product and triggers a second import pass after the initial completion (testing if the target exists later in the same file). If the reference target does not exist in the system at the time of the second import pass, the background process logs an error due to a missing reference target. A second import pass caused by missing targets bears a heavy performance impact.

This type of error illustrates the importance of the sequence of nodes in the import file. Refer to the **Avoid Forward Declarations** section below.

Avoid Forward Declarations

The structure of the import file can be optimized for import performance by ensuring that referenced nodes are not being created in the file after they are needed (forward declaration).

If the reference target exists later in the same import file, then the target does not exist at the time of import. The importer skips the product and triggers a second import pass after the initial completion. The reference target will exist in the system at the time of the second import pass, which will allow the import to succeed.

A second import pass caused by missing targets bears a heavy performance impact. This illustrates the importance of the sequence of the nodes in the import file. Refer to the **Avoid Missing Reference Targets** section above.

Avoid Multiple Updates of Same Object

The structure of the import file can be optimized for maximum import performance by ensuring that the same product or entity is not updated several times in a single file or in multiple files.

- When creating import files, structure the files by modified nodes, not by attribute. It is faster to update a single node with 10 attributes once, than to update each attribute individually.
- Update the same node with all relevant information in one file, rather than splitting it over multiple files or imports. Thus, understand the data patterns of busy endpoints and attempt to structure import files so that any given product or entity appears in the fewest number of import files as possible.

  If a given product or entity is distributed across several import files instead of consolidated into one, this will cause duplicate execution of business rules, approvals and cache load. It also has the potential to dramatically increase revision history, which is a performance detriment of its own.

  Updating the same product or entity at the same time across multiple imports can result in Optimistic Lock failures.

Avoid Unnecessary Business Rules on Import

Using business rules on imports is a powerful tool, but like all powerful tools comes with some risks. Complex JavaScript business actions add execution time to imports.

Review the following items to ensure business rules are used efficiently:

- Are automated actions *required* on import? The fewer automated actions on import, the faster the import performs.
- Does the business process *require* actions on import? When possible, design your solution so imports can run free of any business actions. An event processor can be used to perform business actions that are not necessary to be run on import.
- Are the business rules running on import via workflows or approvals *required*? While extremely useful, all of these options impact performance.

Once it is determined that a business rule is required, verify that there are no unnecessary business actions being performed, such as:

- Are there duplicate executions? For example, an approval triggered in a global business rule as well as in a local state transition rule.
- Are business rules reading or updating objects other than the one being imported (reference sources or targets, parents, children, etc.)? This reduces performance because only the imported product is likely to be resident in cache at the time. Business rules that iterate through children of the imported product are particularly common but very expensive performance-wise.
- Do the business conditions allow a logical exception or lock contention? Verify business rule execution for the purpose of automation or transformation is secondary and is fail tolerant. This means that the criticality of a rule failure is far less than that of the import operation itself. In the event of an exception, rules can be fixed and executed again, whereas import records that are skipped can be difficult to rectify on a busy system.

For more information, refer to the **Business Rule Recommendations** topic.

Avoid Optimistic Locking in Business Rules on Import

If business rules on import are determined to be required, minimize the risk of optimistic locking caused by business actions on import:

- Business actions that execute on tree structures or commonly referenced objects may contribute to optimistic locking or even deadlock failures.
- Any business action that triggers an approval inside of a try / catch block can cause an optimistic lock (as well as subsequent retries and eventual failure) if an approval condition failure is caught and the original exception is not re-thrown.
- Any action that throws an error with the word 'exception' in the error message results in the importer functioning as if an optimistic lock occurred, resulting in retries and eventual failure.

For more information, refer to the **Business Rule Recommendations** topic and the **Optimistic and Pessimistic Locking Recommendations** topic.

Avoid Complex Privileges on Import

When importing, all privileges are checked for each piece of information imported, even when the data is not imported. For example, when performing a hierarchy check (e.g., is the product below a certain root node), performance can be impacted significantly for large imports.

To avoid excessive privilege-checking and improve performance dramatically, the import user should have as broad and as few privileges as possible. For IIEPs, this applies only to the user configured on the endpoint.

Importing for Migration

The import recommendations apply to operational scenarios on production environments. However, the recommendations can also be applied to migration scenarios, although the migration scenarios are usually performed on a separate environment where the results are copied to a production environment via database copies.

Initial data migration is typically handled differently from standard imports because it is a one-time operation and the volume of data is generally far greater than a typical import would be expected to process. It is also generally expected that a greater level of effort will be invested in preparing the import messages or files so that the migration can be completed over a reasonable period.

When preparing migration data files consider the following:

- Transformation business rules may be required specifically for migration. Attempt to avoid using rules that read from or write to many related objects or children. Business rules should, wherever possible, only transform data on the object being imported.
- With serial endpoints, attempt to load products in the smallest number of import files possible. For example, load each product exactly once, providing full attribution. This reduces the number of times each product must read / flush from cache.
- If necessary, set the migration endpoint to parallel processing to use multiple concurrent background processes.
- If there are many references between objects, optimistic locks or deadlocks may occur. Even in the absence of actual locking errors in the logs, you may find that performance is slower than expected due to the lock waits required to update reference targets.
- Consider using two passes. The first set of import files is loaded for an IIEP running parallel and contain all information about the products except for references. The second set of import files is loaded for a separate serialized IIEP and contain only references. Start the product information import IIEP first and allow it to progress enough to load the products being referenced by the second set of files. Both endpoints can run at the same time as long as the products are not being processed by the two IIEPs at the same time.
- When using parallelized endpoints, avoid business rules that update products other than the one being loaded. This scenario can produce optimistic locking errors. If this type of business rule is required, consider using bulk updates to execute the logic after import is complete.

Matching and Linking Recommendations

This is one of the data gathering methodologies and recommendations for functional performance improvement. The full list is defined in the **Performance Recommendations** topic.

Matching and Linking functionality relies on these underlying components:

- **Match Codes**: define a string of attributes to compare, such as name, address, birth date, and email for customer data; or GTIN, EAN, part number, and SAP entity for product data.
- **Matching Algorithm**: evaluates match codes for potential duplicates and displays a percentage to indicate if they are the same record (via auto threshold) or likely the same record (requiring clerical review). The matching algorithm also defines how the duplicates are handled, such as merge or generate a golden record. Survivorship rules of the matching algorithm determine which attributes are copied to the golden record (name, attributes, references).
- **Event Processor**: triggers events which can be acted upon by the matching and linking business rules or matching and linking outbound integration endpoint.

Match codes limit the number of comparisons that the matching algorithm must make, and an event processor launches the associated business rule or OIEP. The system load of matching and linking can have a negative effect on the performance of STEP, especially if there are a lot of supplier items imported, if the matching and linking process is complex, or if the approval of the golden record is part of the process. For more information, refer to the **Matching, Linking, and Merging** documentation.

While the matching and linking and golden record functionalities are designed and based on business requirements, use the following recommendations when a matching and linking process performs poorly.

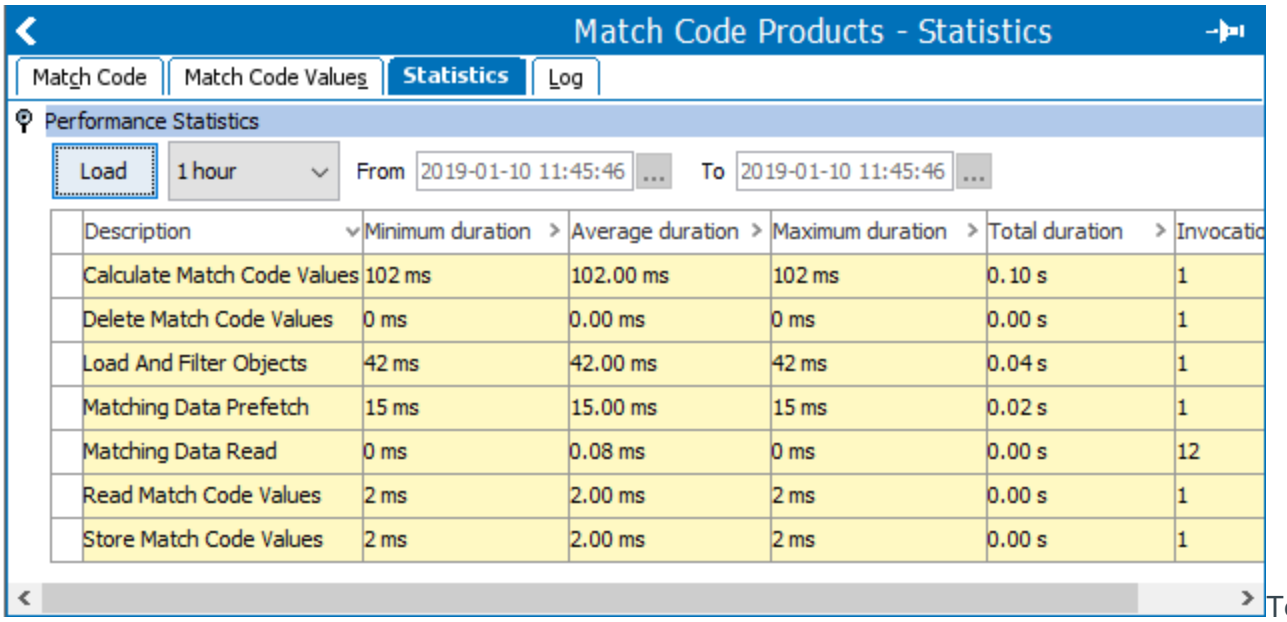
Optimize Match Codes and Matching Algorithm

For details on match codes and matching algorithms, refer to the **Match Codes** section within the **Matching, Linking, and Merging** documentation.

- Evaluate the performance of match code generation via the 'Maintain Match Code Values' option by reviewing the Match Code object's Statistics tab 'Performance Statistics' flipper details. Modify the items reported to have poor performance.
- Evaluate the performance of the matching algorithm execution via the 'Generate Events for Matching' option by reviewing the Matching Algorithm object's Statistics tab 'Performance Statistics' flipper details. Modify the items reported to have poor performance.

Statistics on Match Codes Generation

The performance of the generation of the Match Codes can be measured in STEP. The Match Code Statistics tab allows you to verify the Match Codes are performing well and identify performance problems.

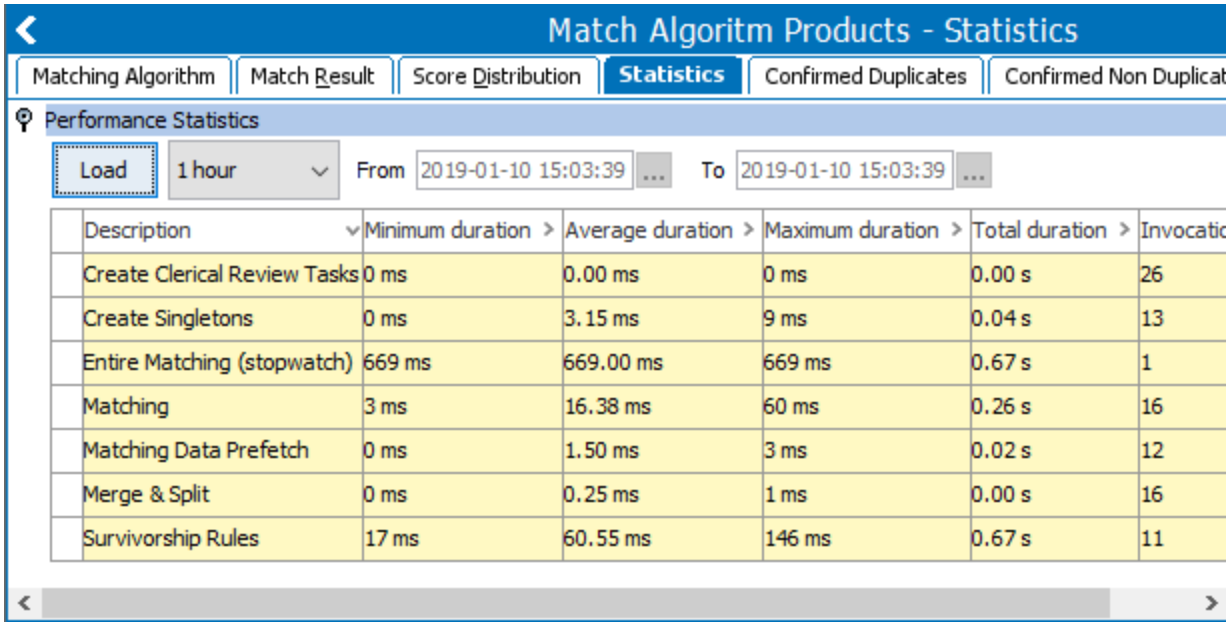


Te

For detailed information on the 'Maintain Match Code Values' option and the 'Match Code Values' tab, start by going to the **Match Codes** section within the **Matching, Linking, and Merging** documentation.

Statistics on Running the Matching Algorithm

The performance of the running of the Matching Algorithm can be measured in STEP. The Matching Algorithm Statistics tab allows you to verify that Matching Algorithm generation is performing well and identify performance problems.



Limit Attributes Promoted to Golden Records

Attributes can be promoted to golden records via the 'Trusted Source Value' survivorship rule. When the attributes belong to an attribute group with thousands of attributes (or more), such as an ETIM attribute group, the golden record process can degrade performance significantly. To avoid this performance issue, limit the number of attributes in the attribute group that is subject for promotion.

In most cases, this can be achieved by defining an attribute group that includes a limited set of attributes.

If the number of attributes in the attribute group cannot be limited (such as ETIM attribute groups), use a business action during the survivorship rules, instead of using the 'Trusted Source' option.

Approve Golden Records Outside of Matching and Linking

Golden records can be approved within the matching and linking process, separate from the process, or not approved at all.

When approving the golden record is required, for recommended performance, use an event processor trigger to run a bulk update after the matching and linking process is complete.

Avoid Multi-Context Survivorship Rules

STEP allows multi-context survivorship rules for context specific names, context specific attribute values, context specific image references, and context specific web-hierarchy links.

However, when a multi-context survivorship rule is executed and the global value is not available, the matching and linking process continues through all contexts to find an attribute value in a local context. Ultimately, the local value is copied to the golden record in that local context. This process negatively effects performance.

In legacy STEP versions, the multi-context survivorship rules (available via a custom extension) were called 'Merging Contexts' as shown below.

Survivorship Rules	
	Criterion
>	Name: Trusted Source Name (Merging Contexts)
>	Golden record kramp only promotion: Trusted Source Value (Merging Contexts)
>	Golden record all source promotion: Trusted Source Value (Merging Contexts)
>	007 - Technical information: Trusted Source Value (Merging Contexts)

In current versions, multi-context survivorship rules are standard and are called 'Multi Context' as shown below.

Multi Context Trusted Source Value

If multiple languages are defined, then a multi-context survivorship rule for a language-dependent name, attribute values, primary image references, or web-hierarchy links attempts to promote the correct value of the supplier record in the right language of golden record.

By default the 'Allow values from …' checkbox is selected, and allows sub-dimension point to be updated from less trusted sources.

The following examples illustrate this effect:

Assume the name is language dependent and the trusted sources are 'SLRV' and 'CNET,' in that order.

- The SLRV source (silver record) only has a name for 'Language Root' context.
- The CNET source has a name for 'English' context (which is a sub-dimension point to 'Language Root').

The English name will only be allowed to be promoted to the golden record if the checkbox is checked.

Optimistic and Pessimistic Locking Recommendations

This is one of the data gathering methodologies and recommendations for functional performance improvement. The full list is defined in the **Performance Recommendations** topic.

STEP uses locks to prevent data corruption due to concurrent updates by multiple processes. For example, consider a bank account where one person deposits $100 and another person concurrently withdraws $100. If the two (2) processes read the balance at the same time, then add or subtract $100 and update the balance, the balance is either increased or decreased by $100. But the correct result is for the balance to remain the same.

In STEP, when concurrent threads attempt to generate new node revisions, locks protect the node's revision history to avoid data corruption. Contention for row locks indicate a thread is locked (often by business logic) while waiting for another thread to commit data. To avoid row lock contention, business logic that runs updates (like bulk updates, approvals, workflow transitions, imports, etc.) should be fast to execute and limit the updates to a small number of objects. For example, updating a field on child products when approving a product may work reasonably well when updating a product with 10 children. However, it is not uncommon for some products to have thousands of children. With the 'update child field' business logic, the approval would likely run for a very long time, and as a result, the product being approved would be locked, blocking other attempts to update that product until the approval operation is complete.

The available locking options include:

- **Optimistic** - This is the default option for STEP and involves no locking but increases the version number of an object in the database. When a process attempts to update a value on a node, it reads the version number, updates the value, and then re-reads the version number. If another process has updated the value in the meantime, then the first process retries up to 10 times (including 2000 ms waiting time) until successful.

  Optimistic locking exceptions indicate that additional attempts were required before the update was successful. The exceptions prevent two (2) identical updates from starting concurrently which would result in an infinite loop. However, even with a successful update, the update runs slowly.

  - Success / Failure: If the version number is the same as it was before the update, the version number is incremented. If, in the meantime, another process has updated the value, an optimistic locking exception is thrown and the update is not committed.
  - User Interface / Automatic Process: If an optimistic locking error occurs in the user interface, the user is immediately notified that another process has already updated the same data. If an optimistic locking error occurs in an automatic process, the lock is reported after the process retries 10 times. The result is that optimistic locking errors can slow down processes without reporting any errors. The longer a transaction takes, the higher the probability of optimistic locking issues when other processes run simultaneously.
- **Pessimistic** - Locking occurs when two or more processes attempt to update a value in the database at exactly the same time. Other processes must wait for the lock to be released before proceeding which can degrade performance. STEP uses pessimistic locking when creating a new revision of a node and when

creating a link to a node unless the target node type is set to a relaxed locking policy.

The disadvantage to pessimistic locking is that until the lock is released, the locked data cannot be edited even if the user that originally locked the row has completed their task. There is also a higher risk for deadlocks.

Deadlocks

Database deadlocks occur when two or more processes have locked a resource, and each process requests a lock on a resource that another process has already locked. One or more transactions are waiting for one another to release a lock on their process.

For example, Process A has resource 1 locked and is requesting a lock on resource 2. Process B has resource 2 locked and is requesting a lock on resource 1. A deadlock occurs and is not resolved until one of the processes releases a lock. Although this example is simplistic, it shows how deadlocks work and can be resolved.

This topic includes the following sections:

- 'Reference Target Lock Policy' Parameter
- Optimistic Locking - Log Level Setting
- Optimistic Locking - Analyze Failures
- Optimistic and Pessimistic Locking Recommendations

'Reference Target Lock Policy' Parameter

Both optimistic and pessimistic locking are impacted by the 'Reference Target Lock Policy' parameter on the object types of the objects being handled. For automated processes that create references to target objects, the 'Relaxed' setting is beneficial on the Reference Target Lock Policy parameter of the target object types.

The 'Reference Target Lock Policy' parameter for an asset, entity, classification, and product object type manages how objects are locked while they are being referenced.

When long-running transactions cannot be simplified easily, consider setting the 'Reference Target Lock Policy' parameter to 'Relaxed' on the object types to which the long transaction applies.

Strict or Relaxed

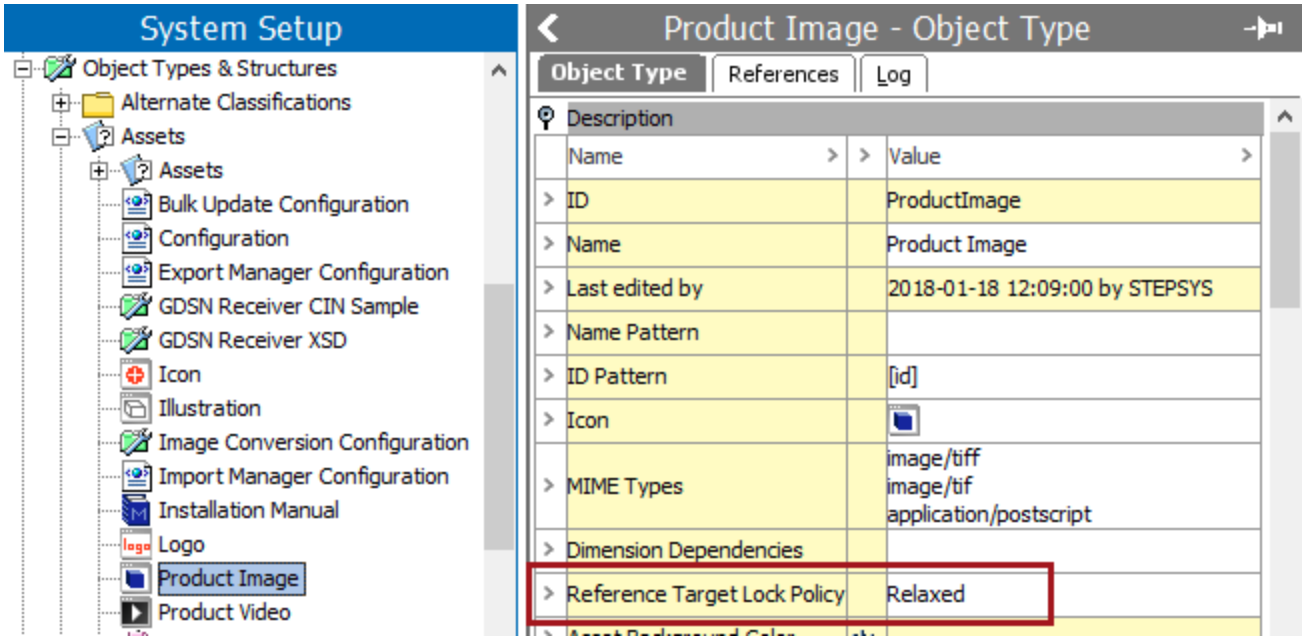
The 'Strict' setting is default for the 'Reference Target Lock Policy' parameter. When a reference between two objects is being created, the target object is locked to ensure that it is not being deleted while the reference is being created. This means that only one process or user can edit it at a time. Often, deletion of the object is not a deletion from the database, but a deletion due to revision control when the insert of a new history entry occurs.

When using the Optimistic Locking option, a 'Strict' setting can result in an issue running parallel imports where the first import locks the object type being referenced, and the second import eventually stops running because it cannot access the locked object. Since STEP continues to retry the import, this can negatively impact inbound feeds.

The 'Relaxed' setting uses a less restrictive lock (a shared 'user lock') on the reference target objects being edited so that concurrent updates by multiple processes and/or users are allowed. A full lock is used only when a deletion is attempted. The Relaxed setting allows faster parallel creation of references to the same target object. This setting, when used with object types that are frequently referenced but rarely deleted, improves the performance and stability of parallel inbound imports, bulk updates, and users concurrently creating references to the same objects.

Important: Do not use the 'Relaxed' setting on an object type for objects that are often deleted as this can result in poor performance, and risks locks and deadlocks.

For example, the image below shows the 'Reference Target Lock Policy' is set to 'Relaxed' for object type 'Product Image (ProductImage)' since the many of the optimistic locking errors in the asset push apply to this object type.



Optimistic Locking - Log Level Setting

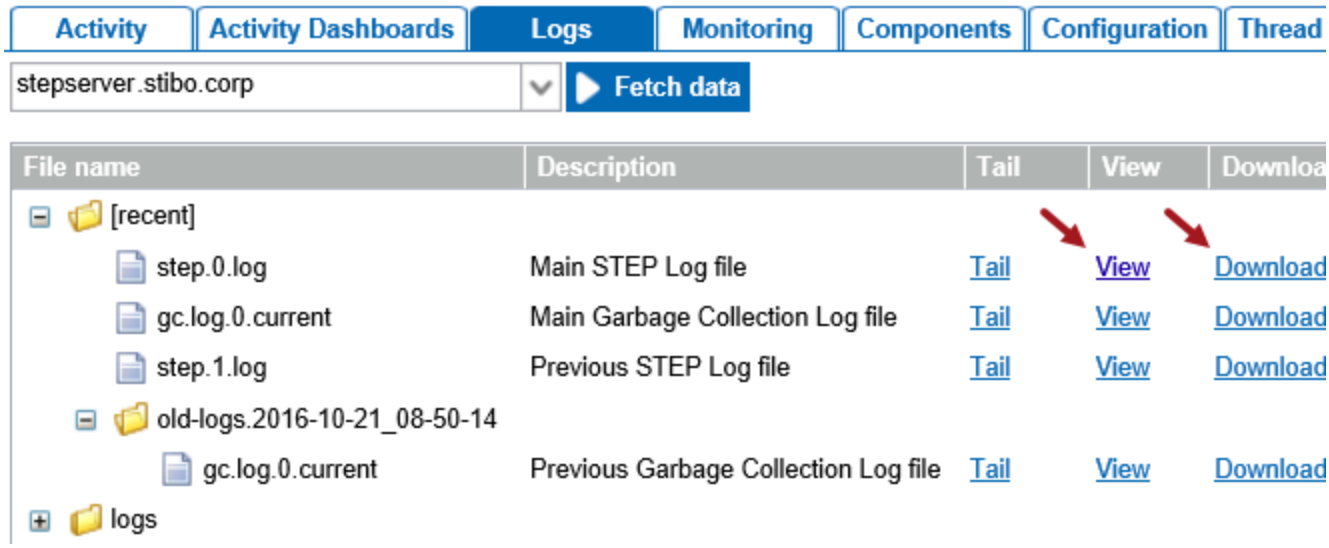
STEP puts a transaction on hold when optimistic locking occurs and after some time, tries to process the transaction again. This illustrates how optimistic locking errors can degrade the performance of imports, exports, asset push, business rules, etc. and can degrade the performance of the STEP system.

The message **Optimistic lock exception occurred, retrying** occurs for optimistic locking errors with fewer than 10 retries. When the **Log.Level.com.stibo.core.domain.impl.ManagerImpl** property is set to FINER or FINEST in the sharedconfig.properties file on the application server, these errors are written to the step.log files. For more information, refer to the **Logs** topic in the **Administration Portal** documentation.

Optimistic Locking - Analyzing Failures

Use the STEP Admin Portal to analyze optimistic locking errors.

1. From the Start Page, click the STEP System Administration button and supply the login credentials.
2. On the **Logs** tab, click the **Fetch data** button to load the data.
3. Select the **Main STEP Log File** and click either the **View** link or the **Download** link to review the log.



4. Search the log for the instance of 'optimistic locking' text.

```
2017/10/23-09:52:10     6d7b|R3404828|PRT|c4userportal  com.stibo.core.domain.impl.state.scxmlimpl.StateFlowImpl evaluateConditionNoThrow WARNING
Business Condition failed for item "C4A-1957171" in state "ModeCommerce" : Wrapped kodo.util.OptimisticVerificationException: Optimistic locking
2017/10/23-09:52:10     6d7b||PRT|c4userportal  com.stibo.portal.engine.server.util.ExceptionConverter convertRuntimeException SEVERE: RuntimeExc
```

5. Determine the cause of the optimistic locking error as shown in the **Examples** section below.
6. Follow the steps in the **Optimistic Locking Recommendations** section to reduce optimistic locking errors.

Examples

Optimistic locking errors are caused by long transactions and may be an import, export, business rule, asset push, etc. The following examples illustrate how to identify what caused the optimistic locking errors.

- Warning for optimistic locking caused by Asset Push

```
2017/10/25-16:34:07 9e8e com.stibo.services.assetpush.beans.AbstractServiceBean
wrapUnexpectedException
WARNING Caught unexpected: kodo.util.OptimisticVerificationException: Optimistic
locking errors were
detected when flushing to the data store. This indicates that some objects were
concurrently modified
in another transaction. Failed objects: [AssetPO@7caa505: 3615004291612_1,
AssetPO@526cb477:
3615004302998_3, AssetPO@3c0fccc3: 3615004302998_pantone, AssetPO@3bd3211d:
3615004291612_3,
AssetPO@696359ef: 3615004302998_1, AssetPO@5e9b87b8: 3615004291612_pantone,
```

```
AssetPO@1cb0217c:
3615004302998_2, AssetPO@3e49203e: 3615004291612_2] [java.util.ArrayList]
```

- Severe optimistic locking error caused by a Web UI transaction in Web UI 'C4userportal'

```
2017/10/17-12:34:59 795|L3022356|PRT|C4userportal
com.stibo.portal.engine.server.util.ExceptionConverter
convertExceptionSerializeAndLocalize SEVERE Type: Error no esperado, Message:
Optimistic locking errors
were detected when flushing to the data store. This indicates that some objects
were concurrently
modified in another transaction. Failed objects: [ProductPO@21d744cc: C4A-305195,
Value7PO@5b8b4f21,
valno: 231765818, qualifier: -5, rev: 0, Value7PO@344dc391, valno: 231765817,
qualifier: -5, rev: 0,
Value7PO@3f2b8947, valno: 231765815, qualifier: -5, rev: 0, Value7PO@5478ad23,
valno: 231765816,
qualifier: -5, rev: 0, com.stibo.core.persistence.NodeStatePO@36bc02a7]
[java.util.ArrayList]
```

- Warning for optimistic locking caused by business condition transaction in state 'ModeCommerce' in Web UI 'C4userprotal'

```
2017/10/23-09:52:10 6d7b|R3404828|PRT|c4userportal
com.stibo.core.domain.impl.state.scxmlimpl.StateFlowImpl
evaluateConditionNoThrow WARNING Business Condition failed for item "C4A-1957171"
in state
"ModeCommerce" : Wrapped kodo.util.OptimisticVerificationException: Optimistic
locking errors were
detected when flushing to the data store. This indicates that some objects were
concurrently modified
in another transaction. Failed objects: [ProductPO@3c0b71a7: C4V-1957172,
ProductPO@2f9030b2:
C4V-1038657, ProductProductReferencePO@284a0999, edgeid: 23130110, rev: 0]
[java.util.ArrayList]
(Script#305) in Script at line number 25 at column number 0
```

Optimistic and Pessimistic Locking Recommendations

For both locking options:

- **Modify the reference target lock policy** - set the parameter to 'Relaxed' when all of these points are true:
  - optimistic locking failures occur frequently in the log file
  - transactions cannot be optimized
  - the object types that are frequently referenced are rarely deleted

  For more information, refer to the **'Reference Target Lock Policy' Parameter** section.

For the optimistic locking option:

- **Optimize large transactions** - keep transactions small to limit the probability of introducing an optimistic locking failure by optimizing the business rules and functions used at e.g., imports, exports, bulk updates.
- **Temporarily set** `Log.Level.com.stibo.core.domain.impl.ManagerImpl = FINER or FINEST` to analyze all optimistic locking issues as defined in the **Optimistic Locking - Log Level Setting** section.
- **Analyze optimistic locking problems** - defined in the **Optimistic Locking - Analyzing Failures** section.

Privilege Recommendations

This is one of the data gathering methodologies and recommendations for functional performance improvement. The full list is defined in the **Performance Recommendations** topic.

While STEP allows a very granular privilege system and privilege setups, complex privilege models can lead to a degradation in performance. Running STEP as a user with a large number of very specific privileges influences the performance of any action in STEP that goes across a large number of nodes, values, or references. This performance impact includes export, import, bulk update, recursive approval, matching, and 'multi views' like task list and multi editor.

For more information, refer to the **Privilege Rules** topic in the **System Setup** documentation.

Privilege Configurations

Privileges are additive only, which means that whenever a basic action is executed, STEP looks for the first privilege that provides the permission.

In terms of performance, the most expensive privilege check is attempting a task for which the user does not have access. The least expensive privilege check is when a user has global permission to everything.

Additionally, consider the following when setting privileges:

- Very specific and granular permissions result in a longer search for the appropriate privilege.
- Assigning privileges on a group of objects (and using the hierarchy to access these objects) provides a less expensive listing than assigning privileges on each object separately.
- Avoid excessive privilege checking to improve performance.

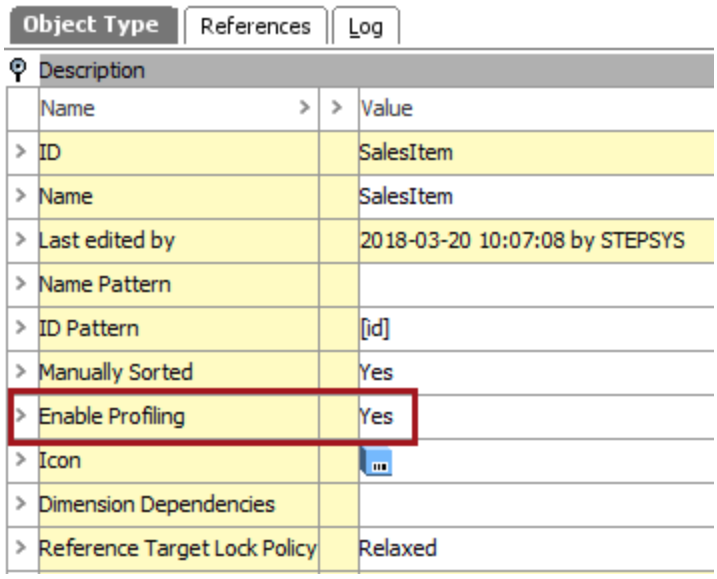
Profiling Recommendations

This is one of the data gathering methodologies and recommendations for functional performance improvement. The full list is defined in the **Performance Recommendations** topic.

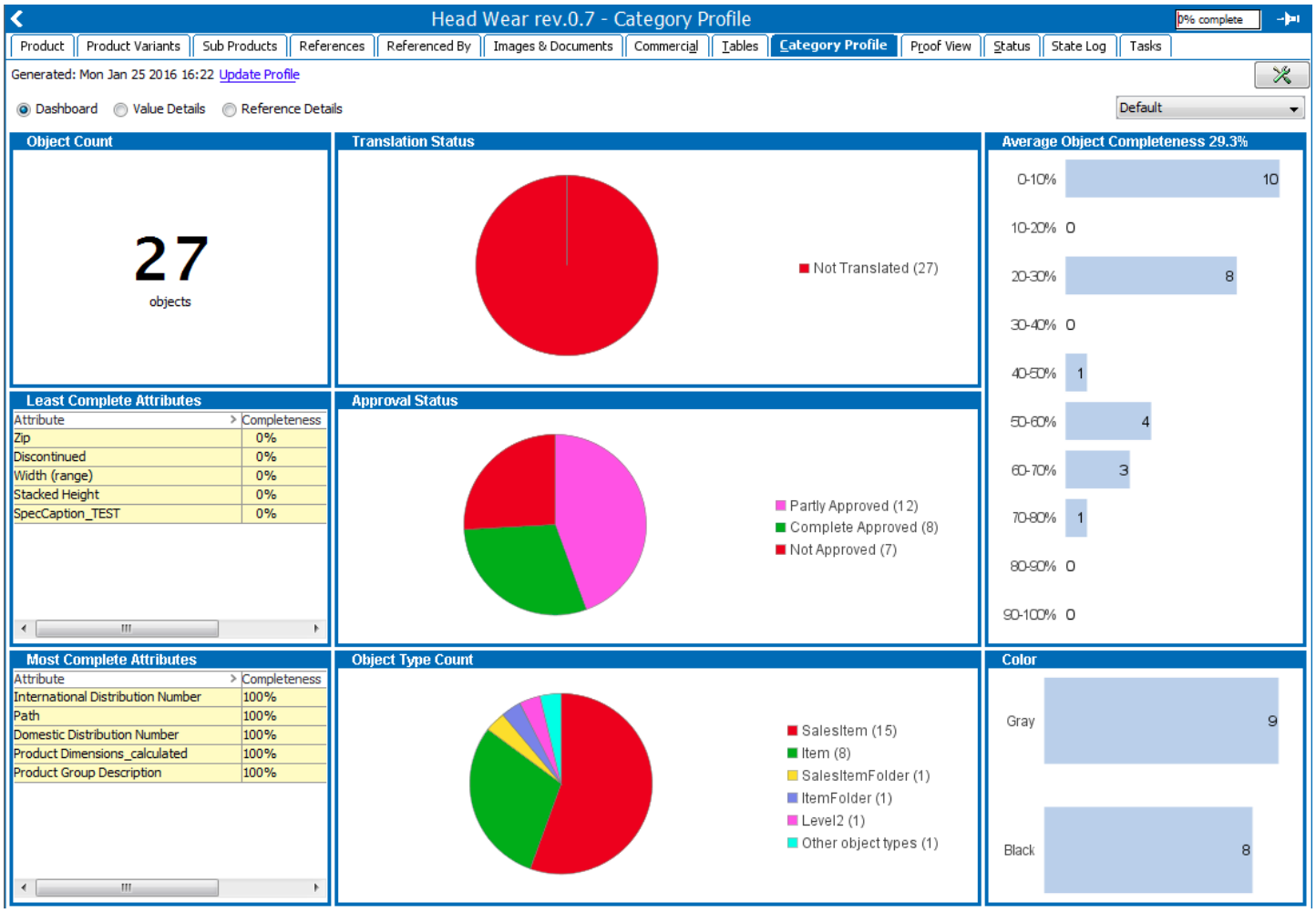
The Category Profile and Data Profile functionality provides a detailed overview of data in a specific branch of the hierarchy in Tree. However, when large categories are profiled, the system uses a lot of memory which can have a negative impact on system performance.

To identify and analyze object types with

Profiling is enabled in System Setup, using the 'Enable Profiling' parameter with the Object Types & Structures node.



When a profile is run, information about the data is displayed as follows and provides access to correct data errors:



For more information, refer to the **Data Profiling** documentation.

Recommendations

Review object types with profiling enabled by exporting all object types using the STEPXML template below and search for 'IsCategory="true".

```
<STEP-ProductInformation>
<UserTypes ExportSize="All"/>
<EdgeTypes/>
<CrossReferenceTypes ExportSize="All"/>
</STEP-ProductInformation>
```

- **Only enable profiling when required**.
- **Limit memory usage when profiling is enabled** via the following case-sensitive sharedconfig.properties:

  DataProfile.MaxDistinctAttributeValuesConsideredDuringProfileGeneration - Sets the maximum number of distinct attribute values per attribute considered during profile generation. The default setting is 100. When the limit is reached, the following happens:

  - Frequent value counts can become inaccurate. STEP uses a counting implementation dedicated for counting in big data collections with a limited memory usage from Clearspring Analytics.
  - The rare value count is disabled because only a frequent count can be maintained. In the profile, the frequent and rare values cells for attributes with too many distinct values are displayed with a light red background color. The attribute completeness and count, and the value instance counts for profiled attributes are correct

  DataProfile.MaxDistinctTargetsConsideredDuringProfileGeneration - Sets the maximum number of distinct targets for the reference or link type that is profiled. The default setting is 100.

- **Consider In-Memory** - Optimizes profiling performance. For more information, refer to the **In-Memory Database Component for STEP** topic of the **Resource Materials** section of online help.

Revision Control Recommendations

This is one of the data gathering methodologies and recommendations for functional performance improvement. The full list is defined in the **Performance Recommendations** topic.

'Revisions' in STEP are historical versions of objects. A revision represents a historical 'snapshot' of an object. Major data objects in STEP (i.e., products, assets, classifications, entities, etc.) are under revision control. This means that previous versions of an object can be viewed, compared, and revived.

For more information, refer to the **Revisions** topic and the **Generating Revisions** topic in the **System Setup** documentation.

Storing many revisions can have a negative impact on system performance. Use the following recommendations to keep the number of revisions under control and to remove unnecessary revisions.

Optimizing performance in revision control involves the following steps and each is defined below:

- Setting the revision threshold
- Maintaining object revisions
- Maintaining integration endpoint revisions

Setting the Revision Threshold

By default, revisions on objects are created when a user makes a change to the object and the number of hours set in the threshold is exceeded (starting when the object was first touched after the last revision was made).

This is particularly useful in cases where an object is primarily maintained by a single user and would only be made by that user choosing to do so manually. The threshold functionality ensures that changes are recorded, without creating an excessive number of revisions, which can have a negative effect on system performance.

In workbench, the revision threshold parameter is in the System Setup 'Users & Groups' node under the Revisability Settings flipper. For details about this threshold, refer to the **Revisability Settings** topic within the **System Setup** documentation.

To avoid excessive creation of the revisions, keep the default setting (168 hours) for the threshold parameter.

Maintaining Object Revisions

As the revision history grows, it impacts the time it takes to retrieve the front revision of the object, including its name, attribute values, and references. This may have a negative effect on the system performance.

To reduce poor system performance due to excessive revisions, define a revision policy as follows:

1. Manually delete revisions that are no longer needed. This creates a baseline for ongoing maintenance.
2. Automatically purge old revisions via one or more event processors to keep the number of revisions under control (i.e., schedule the event process to run monthly, and to delete revisions older than one month).

For details about purging old revisions manually, refer to the **Maintaining Revisions** topic in the **System Setup** documentation.

For details about purging old revisions automatically, refer to the **Creating an Event Processor** topic and the **Revision Management Processing Plugin Parameters and Triggers** topic in the **System Setup** documentation.

Maintaining Integration Endpoints Revisions

Integration endpoints (IIEP and OIEP) with revisions in the thousands (or more) can have a negative impact on system performance and IEPs open very slowly. Typically, this is a result of the IEP poller being started by a different user than the one configured in the IEP.

Identify IEPs with different users

1. Run the Administration Portal Healthcheck for pollers started by a different user than the one configured in the IEP to identify the scenario.
1. From the Start Page, click the **STEP System Administration** button and supply the login credentials.
2. On the **Healthcheck** tab, open the Data Error section and select the 'Pollers started by a different user than the one configured in the IEP' test.
3. Click the Run Selected Tests button (

).
4. When the 'Last Run' column for the test shows today's date / time, review the results in the Detected Problems area at the bottom of the screen.

For more information, refer to the **Healthcheck** topic in the **Administration Portal** documentation.

Update IEPs to resolve different users

1. Log in to the workbench and open an IEP that was reported in the Healthcheck above.
2. Identify the user on the IEP and, if necessary, log in to workbench again as that user.
3. Disable the IEP.
4. Enable the IEP again to set the same user under Revisions (on the Status tab) and the User parameter.

Purge old IEP revisions

After verifying the excessive number of revisions will no longer be created, remove the old IEP revisions that exist in the database.

Use the 'Purge old revisions' option to delete revisions for 'Inbound Integration Endpoint Type' and 'Outbound Integration Endpoint Type' as defined in the **Generating Revisions** topic in the **System Setup** documentation.

Important: Purging a large number of revisions on IEPs can require an index rebuild in the database. Contact your database administrator or your Stibo Systems representative.

Scheduled Process Recommendations

This is one of the data gathering methodologies and recommendations for functional performance improvement. The full list is defined in the **Performance Recommendations** topic.

Scheduled processes can negatively influence system performance if, for example:

- the bulk update, business rule, or search query to fill the collection is not optimized.
- the data profile processes are no longer required by the business.
- the exporter process returns warning or errors.
- the structured translation using Excel or XML checks for translation export requests every few minutes.

Analyzing scheduled processes and resolving the identified issues can potentially improve performance.

For details on scheduling a background process, refer to the **Creating a Scheduled Background Process** topic in the **System Setup** documentation.

Recommendations

Resolve degraded performance with these improvements:

- If an automated schedule is necessary, verify the frequency is warranted. If possible, use the 'Later and Repeat' option to set a recurring schedule to daily (or less frequently), rather than hourly or by minutes.
- Balance the load on the STEP system by coordinating the scheduled processes. If possible, schedule long-running scheduled processes to run sequentially and during system down time.
- Delete scheduled processes without a business value.
- Resolve errors and warnings in scheduled processes (both are visible in the '# of errors' column and also in the log file by searching for the text 'Caught exception handling bg-process').

Optimizing performance in scheduled processes can also include the following:

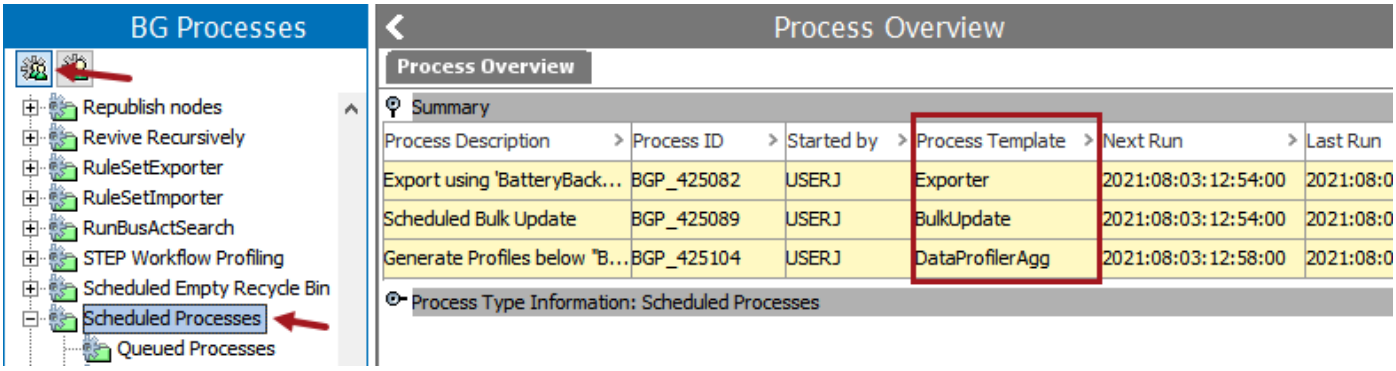
- Structured Translation - Excel or XML, as defined in the **Scheduling a Data Translation** topic of the **Translations** documentation.
- Improve BGP performance using parallel and multi-threading properties, as defined in the **Background Processes and Queues** topic in the **System Setup** documentation.
- Consider In-Memory for scheduled BGPs since it can improve performance particularly with long-running search queries. For more information, refer to the **In-Memory Database Component for STEP** topic in the **Resource Materials** section of online help.

Create an Excel Overview of Scheduled Processes

Review the scheduled processes on your system.

1. In workbench, on the BG Processes tab, click the all users button ( ) to display processes for all users.

2. Expand the **Scheduled Processes** node to view the scheduled processes.

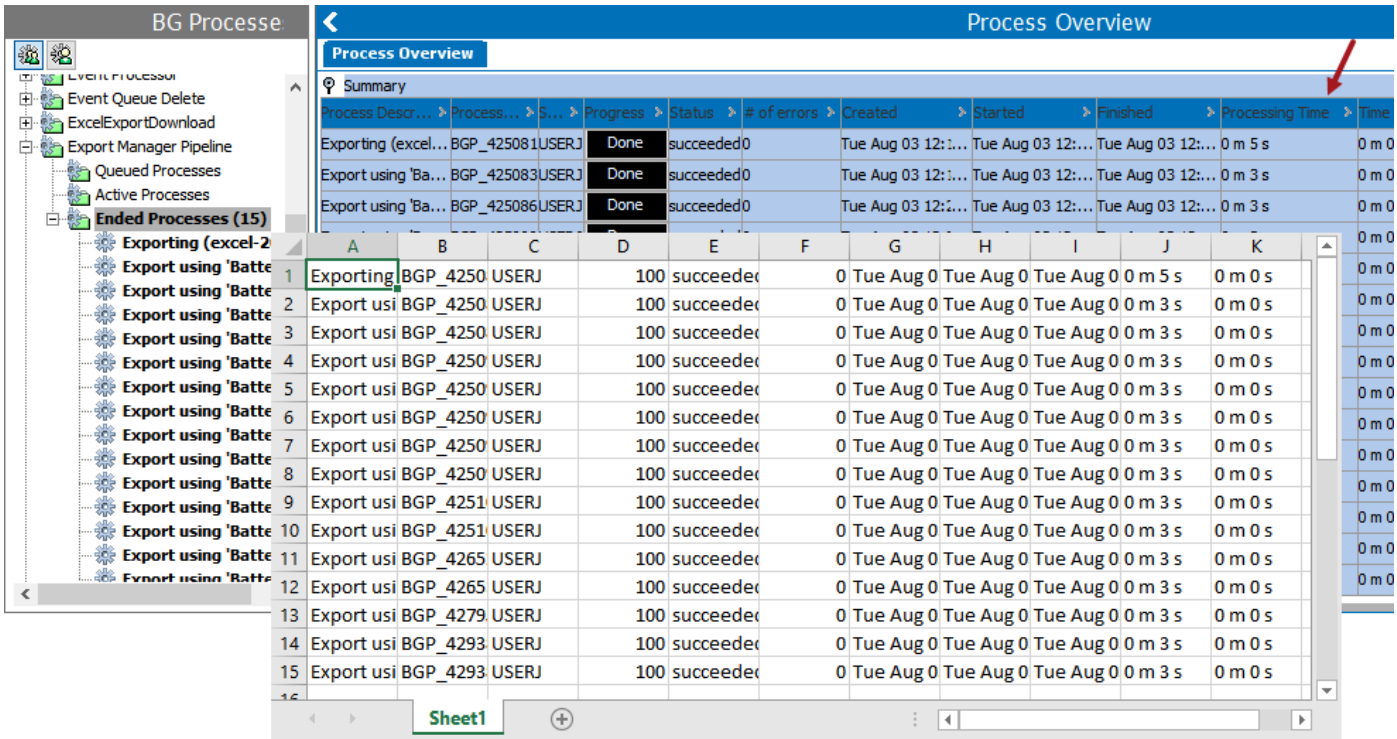


3. For each Process Template entry, copy the scheduled processes summary and paste to Excel:

   - Click the node type on the list of scheduled processes (in this example, the Exporter queue is used by the Export Manager Pipeline, so click the Export Manager Pipeline node, and then repeat for the others) and click the Ended Processes folder.

   - Select all columns in the Process Overview tab in the Summary flipper (including the **Processing Time** column), right-click, and click **Copy** (or use Ctrl+C) to copy the data.

   - Paste the copied process data to Excel and add a column header row.



- Repeat for each type of template.

4. For each Process Template type, follow the steps below to analyze and optimize the scheduled processes.

Analyzing and Optimizing Scheduled Processes

Review the scheduled processes for your system.

1. In the Excel file, review the total number of scheduled processes.
   - Performance issues are likely when 25 or more processes are scheduled to run more often than daily.
2. In the Excel file, review the scheduled processes and verify each is valid for current business requirements.
   - Delete unnecessary scheduled processes as defined in the **Deleting a Scheduled Background Process** topic in **System Setup** documentation.
3. In the Excel file, sort data by the **# of errors** column data to address the scheduled processes that are running with errors and warnings.
   - Resolve the problems identified.
4. In the Excel file, sort data by the **Processing Time** column data to identify long-running scheduled processes.
   - Determine if the scheduled processes are still necessary. Delete processes that no longer provide the intended business value.
   - Continue with the **Analyzing and Optimizing Long-Running Processes** section below.

Analyzing and Optimizing Long-Running Processes

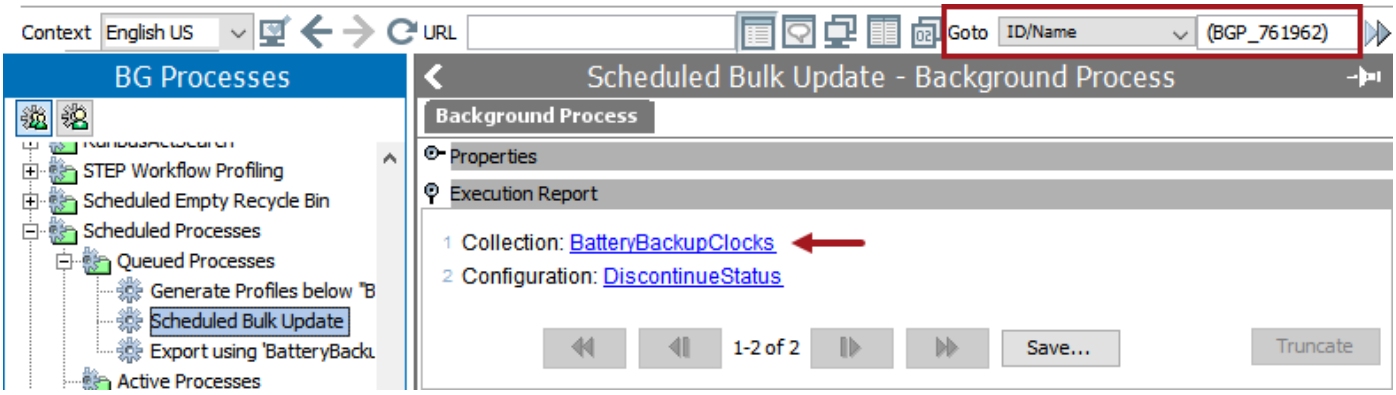
Since repeated searches are typically used in scheduled processes, it is important to analyze the run time of the scheduled processes to identify the searches that might be improved.

> **Note:** The database Oracle KODO cache default limit is 10,000. For example, a hierarchy with more than 10,000 children, a product with more than 10,000 references, a list of value with more than 10,000 values, etc. Storing very large data relations can have great negative performance impacts, since there is a high likelihood that a large number of the related objects are no longer in the cache. For more information, refer to the **Base Setup Recommendations** topic.

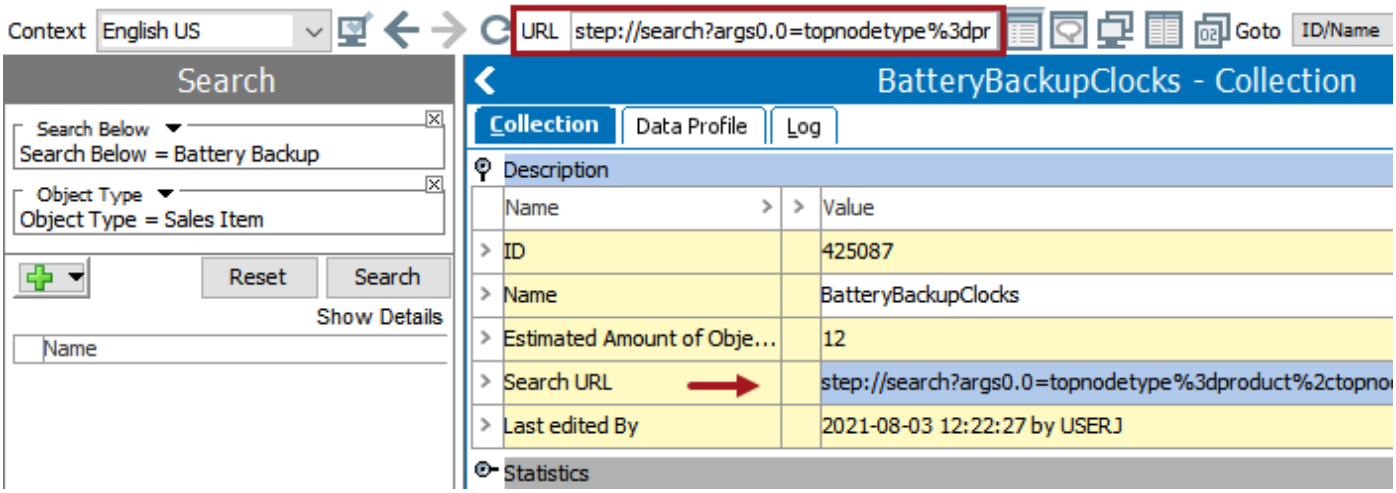
Business rules can also be used in scheduled bulk update processes. Analyze the run time of the scheduled processes and verify which business rules in bulk updates might be subject to improvements.

Review the scheduled long-running processes for your system.

1. In the Excel file, copy the BGP ID for a Scheduled Bulk Update process.
2. In workbench, search for the BGP ID, view the Background Process tab Execution Report flipper, and click the **Collection** link.

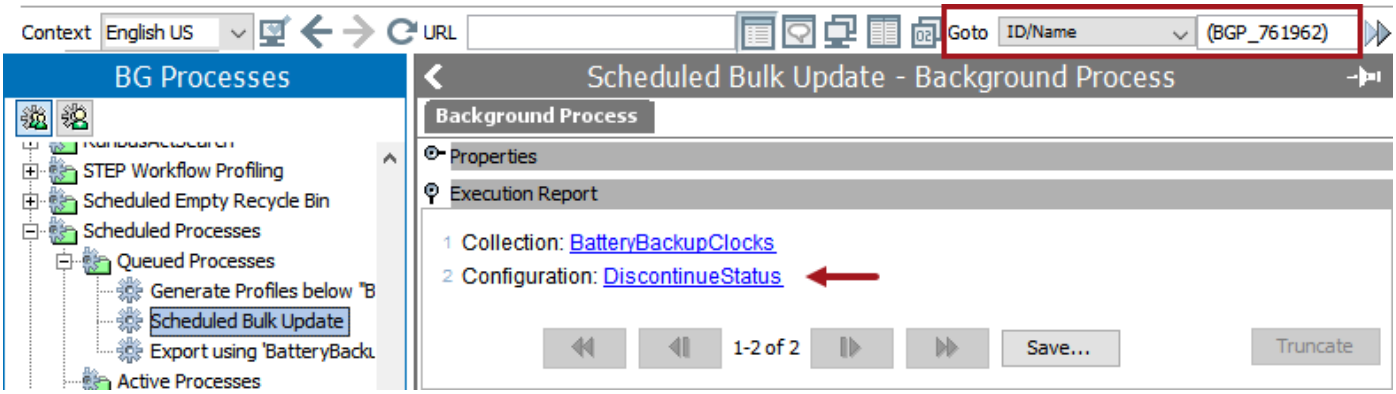


3. If the collection includes a Search URL, copy the **Search URL** parameter value, paste it into the URL parameter, and press **Enter**.

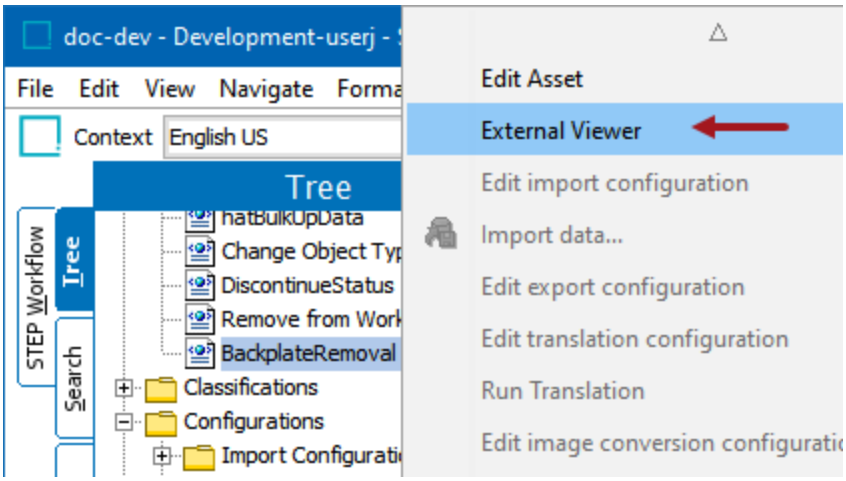


- Click the **Search** button and evaluate the length of time the search takes to respond. For a long-running search
  - If many objects are being searched, use as specific search criteria as possible.
  - Review the **Search Elements to Use** topic or the **Search Elements to Avoid** topic and implement the recommendations.

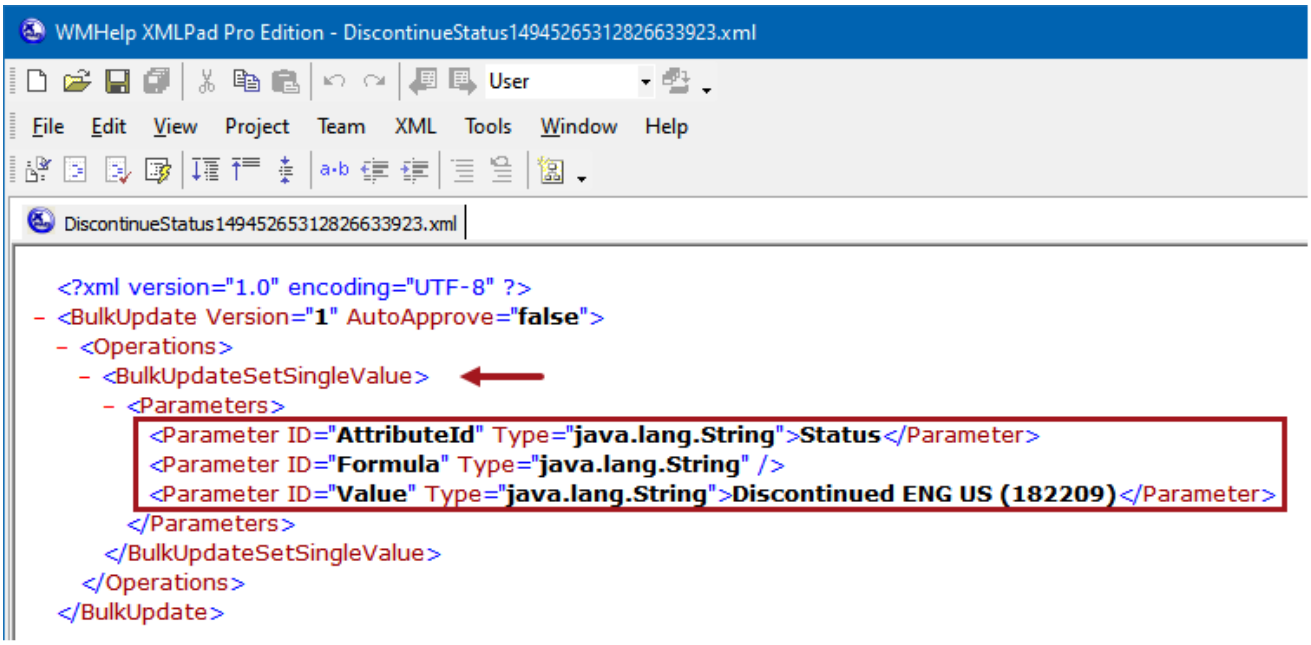
4. Search for the background process again and on the Execution Report flipper click the **Configuration** link.

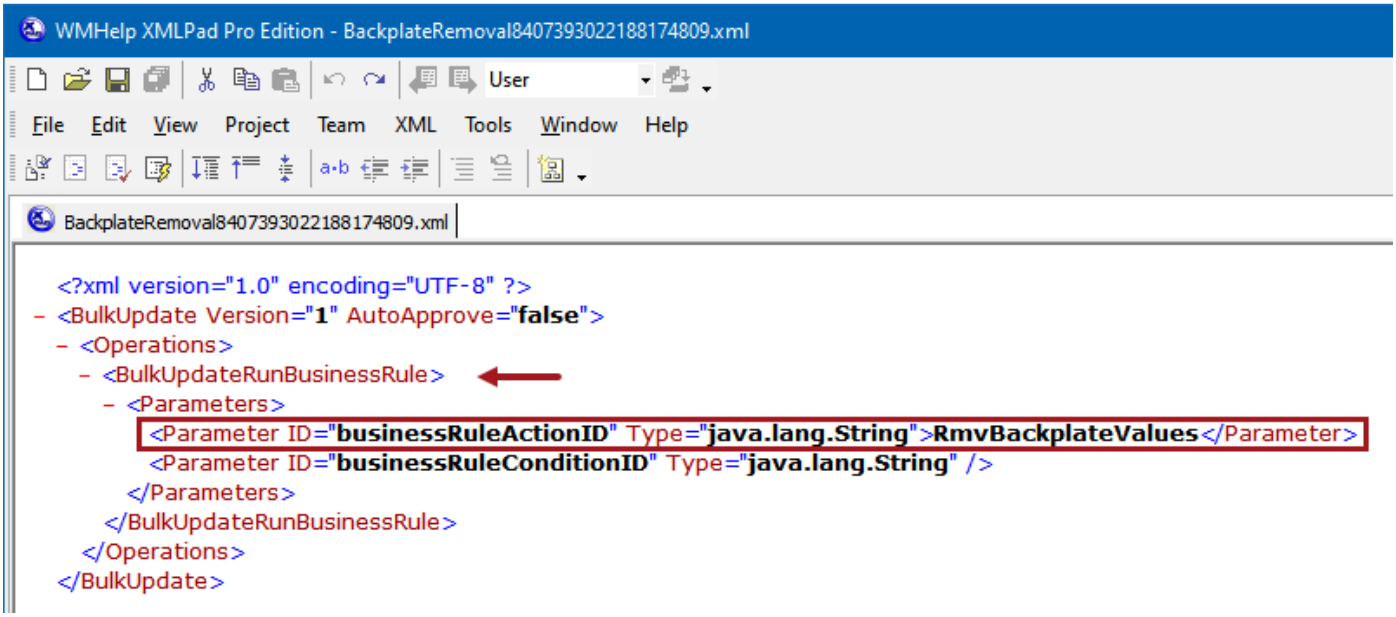


5. Click the Tree tab to show the highlighted configuration, right-click and select **External Viewer**.



6. In the XML viewer displayed, observe the configuration details of the Bulk Update operation. In the examples below, the Set Single Value operation and the Run Business Rule operation are shown below the Operations tag.





- For a **BulkUpdateRunBusinessRule** operation, in workbench, search for the business rule(s) by ID (for example, 'RmvBackplateValues' in the previous image). A business condition can also be included. Open the business rule(s) and analyze the statistics. For details, refer to the **Business Rule Recommendations** topic and the **Business Rule Analysis** topic.

Search Elements to Use

The following list includes search elements known to minimize impact on system performance. This list can be used to troubleshoot existing searches, and can also be reviewed prior to creating new searches to prevent performance problems.

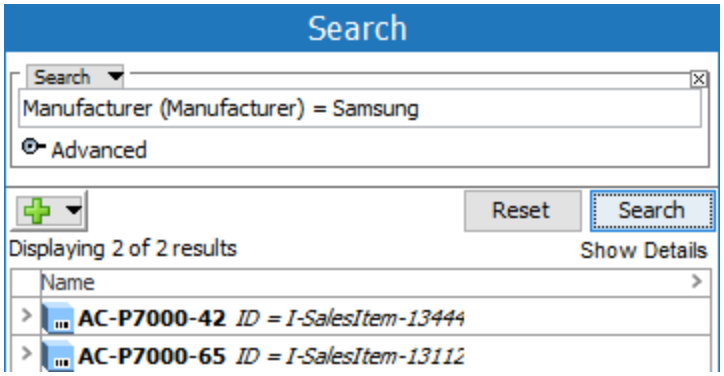
For general information on searches, refer to the **Search** topic in the **Getting Started** documentation.

Basic Searches

When possible, use the following syntax to achieve the best search performance. Searching for a value without specifying the attribute, ID, or name searches for the value in all attribute values, IDs, and names.

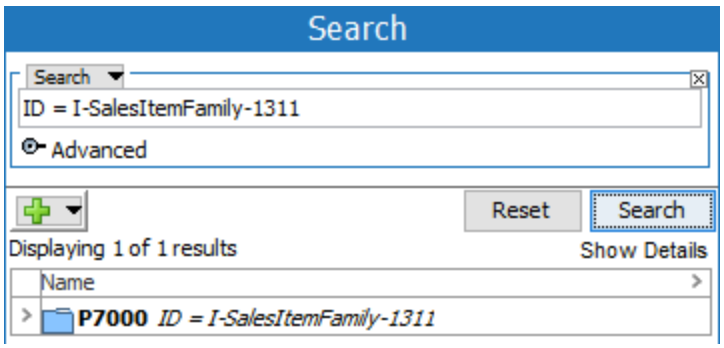
Attributes

Search using 'Attribute-ID = value' to search in values of this one specified attribute only. For example, search for the 'Samsung' value only in the Manufacturer attribute, rather than in all attributes.



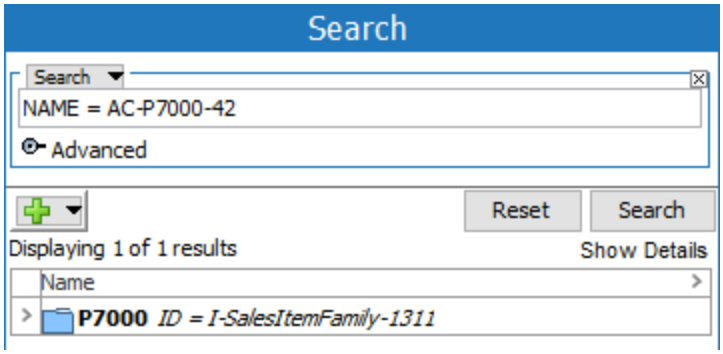
Objects by ID

Search using 'ID = value' to search for objects by looking in the ID parameter only. For example, search for the 'I-SalesItemFamily-1311' value only in the ID field.



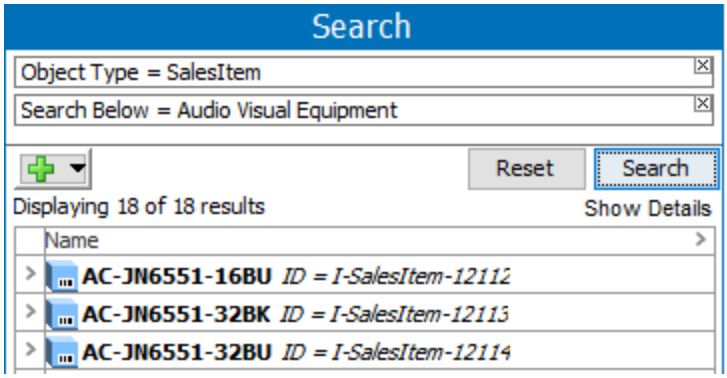
Objects by Name

Search using 'Name = value' to search for objects by looking in the Name parameter only. For example, search for the 'AC-P7000-42' value only in the Name field.



Use Specific Search Criteria

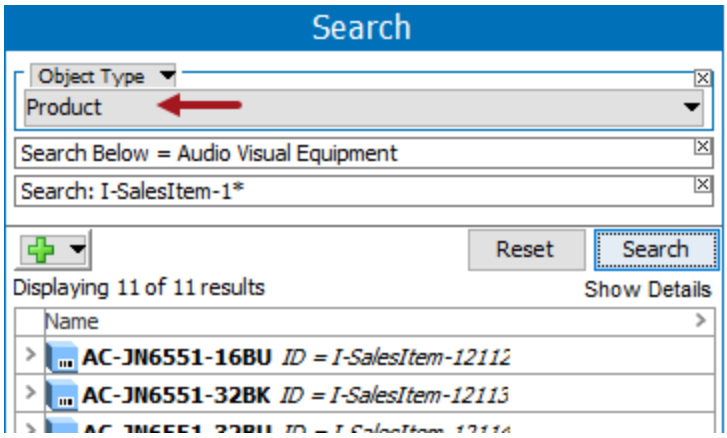
When searching a database with many objects, use as specific a search criteria as possible. For example, specifying the object type to search for in a specific hierarchy (as shown below) is faster than searching for all object types in all hierarchies.



Use Object Super Types

The search function performs slightly better when specifying the object super type (such as product, entity, asset, classification, and so on) instead of using the specific object type below the main object types. For more information, refer to the **Object Super Types** topic in the **Getting Started** documentation.

In the example shown below, the product super object type is used, instead of SalesItem (which is an object type below the product super type).

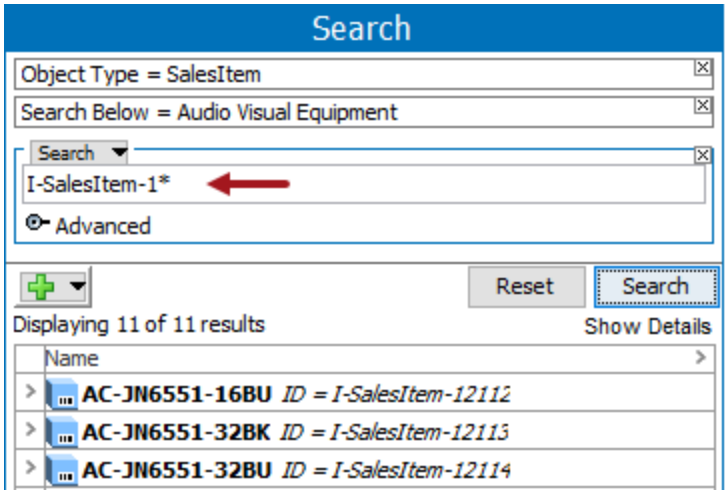


Searches with Wildcards

Specify as many characters as possible before using a wildcard (*) to optimize search performance results.

Important: Specifying a wildcard as first character prevents the use of the database indexing query, which means the search must traverse all objects in the database. This is a dramatic performance impact.

For example, use a wildcard (*) after more than three characters, as shown below.



Optimize Search Below with a Configuration Property

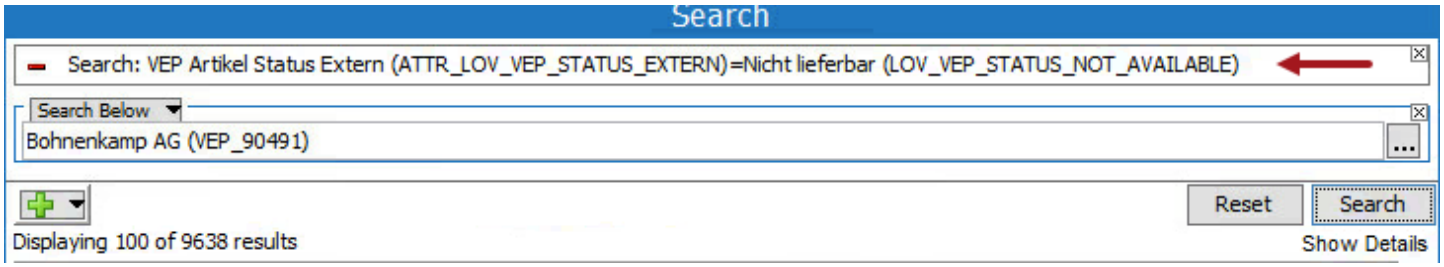
In some cases, a bad execution plan is implemented by Oracle when using the 'Search Below' criteria. The following case-sensitive entry in the sharedconfig.properties file will improve the search performance:

```
Domain.BelowCriteria.UseRecursiveWith=true
```

Setting this property to 'true' forces Oracle to choose another (and in some cases better) exclusion plan, and may have a positive effect on search performance.

Optimize Combined Search Below and Value Search

Combining the 'Search Below' criteria with a value search criteria may not perform well. However, this type of search can be optimized by first doing a value exclusion search, and then combining the result with the 'Search Below' criteria. For example, review the image below:



In-Memory

Consider In-Memory to improve import performance, as defined in the **In-Memory Database Component for STEP** topic in the **Resource Materials** section of online help.

Search Elements to Avoid

The following list includes import elements known to degrade system performance. This list can be used to troubleshoot existing imports, and can also be reviewed prior to creating new imports to prevent performance problems.

For general information on searches, refer to the **Search** topic in the **Getting Started** documentation.

Avoid Inherited Values and Regular Expressions

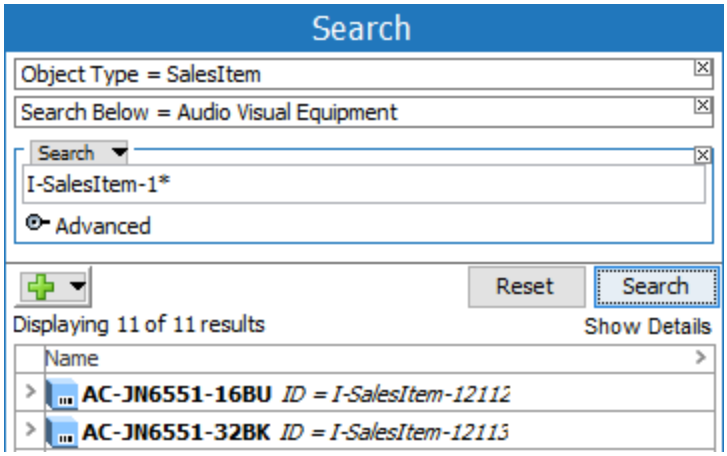
Advanced searches in the workbench and Web UI should avoid the following options to achieve the recommended performance:

- 'Include Inherited Values' - only use when required.
- 'Regular Expression' - disables database indexing, only use when required.

Avoid Root Hierarchies in Search Below

The 'Search Below' option should be used with caution and only when necessary to achieve the required result. Specifying many root nodes in Web UI searches is specifically not recommended.

For example, use a granular hierarchy node instead of searching the full primary product hierarchy.



File Hygiene Recommendations

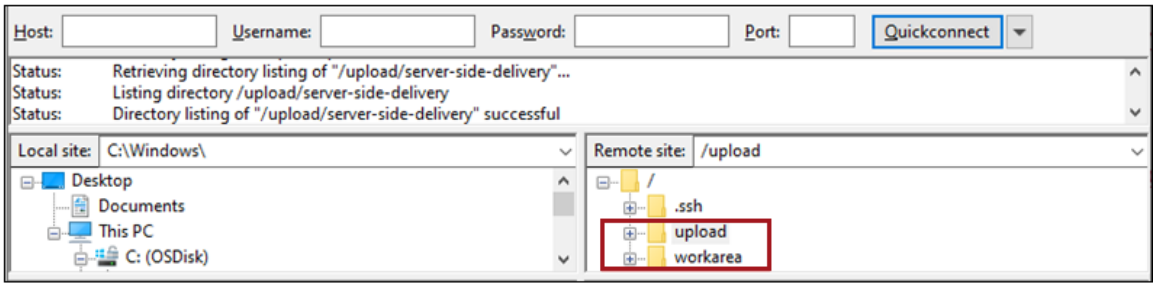
This section describes the data gathering methodologies from a technical viewpoint for the clean-up of folders for hotfolder import and background processing.

Clean Up Import Files

When the STEP application server is overloaded, it can be caused by the number of import files in hotfolders. Furthermore, large import files can also cause the system to be slow.

Analyze the import files situation as follows:

1. Set up an SFTP user in the STEP SaaS Self-Service UI and pay attention to the SFTP URL, which will typically be: `<environment-name>-sftp.mdm.stibosystems.com`.
2. Use your preferred SFTP client application to connect via the SSH key associated with the previously created SFTP user.
3. When connected, you will have folders for 'upload' and 'workarea' (as shown below):



4. Look through the subfolders to find the number and size of import files in the hotfolder location under 'upload.'

Identify the Causes

If the number of import files in the hotfolder is in the hundreds of thousands (e.g., 250k files), or if the size of the import files is too large, continue with the following activities to determine the cause.

- **IIEP 'Keep file after load'**

  Review the active hotfolder and REST Receiver IIEPs and verify that they are configured to automatically clean up the import files. Set the 'Keep file after load' parameter to 'No' to remove import files after success import. For more information, refer to the **Hotfolder Receiver** topic of the **Data Exchange** documentation.

- **IIEP failed or completed with errors**

Review the IIEPs for those that regularly complete with errors, or fail. In these cases, the import process is saved along with the imported file for error tracing. When the import process errors, the accompanying background processes are not removed. This means neither the import files nor the background processes are being removed. When an IIEP successfully completes without errors, both the background processes and the import files can be deleted.

IIEPs that encounter Optimistic Locking errors can result in background processes that fail or 'complete with errors.' In this case, consider setting the 'Reference Target Lock Policy' to 'Relaxed' on the object types for which the long transaction applies. For more information, refer to the **Reference Target Lock Policy on Object Types** topic in the **System Setup** documentation.

- **IIEP non-standard configuration**

IIEPs are created with a standard import directory structure as illustrated below. Modifying that structure can prevent the removal of the imported files after successful import and processing.

Folder	Example of Standard Config	Description
root	/shared/upload/hotfolders/products/	
in	/shared/upload/hotfolders/products/in	import files reside, and are removed after processing
save	/shared/upload/hotfolders/products/save	import files remain when they are imported and when configured to keep the import files after successful processing
error	/shared/upload/hotfolders/products/error	import files remain when the import process completed with errors
failed	/shared/upload/hotfolders/products/failed	import files will remain when the import process failed

Remove Import Files

Based on the reason for the excess import files, follow the steps below to remove them manually:

1. If the import files were not removed because the 'Keep file after load' was previously set to 'Yes,' delete those files from your SFTP client.
2. If the import files were not removed for a reason other than the 'Keep file after load' setting, this indicates the import files are tied to background processes.

   Remove the background processes of the IIEP, which also removes the corresponding import files.

Important: Removing these import files by deleting them directly from the file system will not delete the corresponding background processes.

Create a Background Processes Maintenance Plan

Settings on IEPs and on background process allow configuration of automatic clean-up activities.

IEP Auto Delete Settings

Use the following parameters to configure automatic deletion of background processes on IIEPs and OIEPs:

Inbound Integration Endpoint | Background Processes | Statistics

Configuration	
Pre-Processor	No pre-processing
Process Engine	Asset Importer
Post-Processor	No post-processing
Error Handling & Reporting	Not Defined
Schedule	Not scheduled
Queue for endpoint	InboundQueue
Queue for endpoint processes	In
Transactional settings	None
Maximum number of old processes	100
Maximum age of old processes	1 week

Outbound Integration Endpoint | Configuration | Event Trigge

Configuration	
Process Engine	STEP Exporter
Error Handling & Reporting	Not Defined
Schedule	Start every minute
Queue for endpoint	OutboundQueue
Queue for endpoint processes	Out
Transactional settings	Strict
Number of threads	1
Maximum number of old processes	1000
Maximum age of old processes	1y
Contexts	English US, Global, Spain
Workspace	Approved

- **Maximum number of old processes**: Specify the number of ended processes the system will keep. Succeeded and ended processes are deleted when the number exceeds the specified limit. The oldest processes are deleted first. Setting this number too high may eventually degrade performance.

  For example, if 'Maximum number of old processes' is set to '1000,' a maximum of 1000 succeeded and ended background processes will be retained. The oldest background processes are deleted automatically when the number exceeds the specified limit of 1000.

- **Maximum age of old processes**: Specify the maximum age of ended processes that the system will keep. Ended processes are deleted when the maximum age is exceeded. Setting this number too high may eventually degrade performance.

  For example, if 'Maximum age of old processes' is set to '1 y,' the background processes older than 1 year are deleted automatically.

Ensure the configurations represent a realistic situation as demonstrated below:

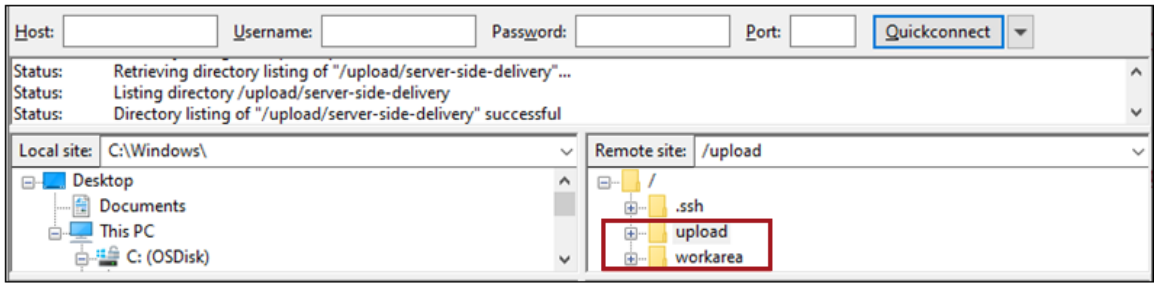
- In an environment with a large number of small imports and exports, use '1000' as a maximum number of old processes and '1 w' as a maximum age of old processes.

- In an environment with small number of large imports and exports, use '50' as a maximum number of old processes and '1 m' as a maximum age of old processes.

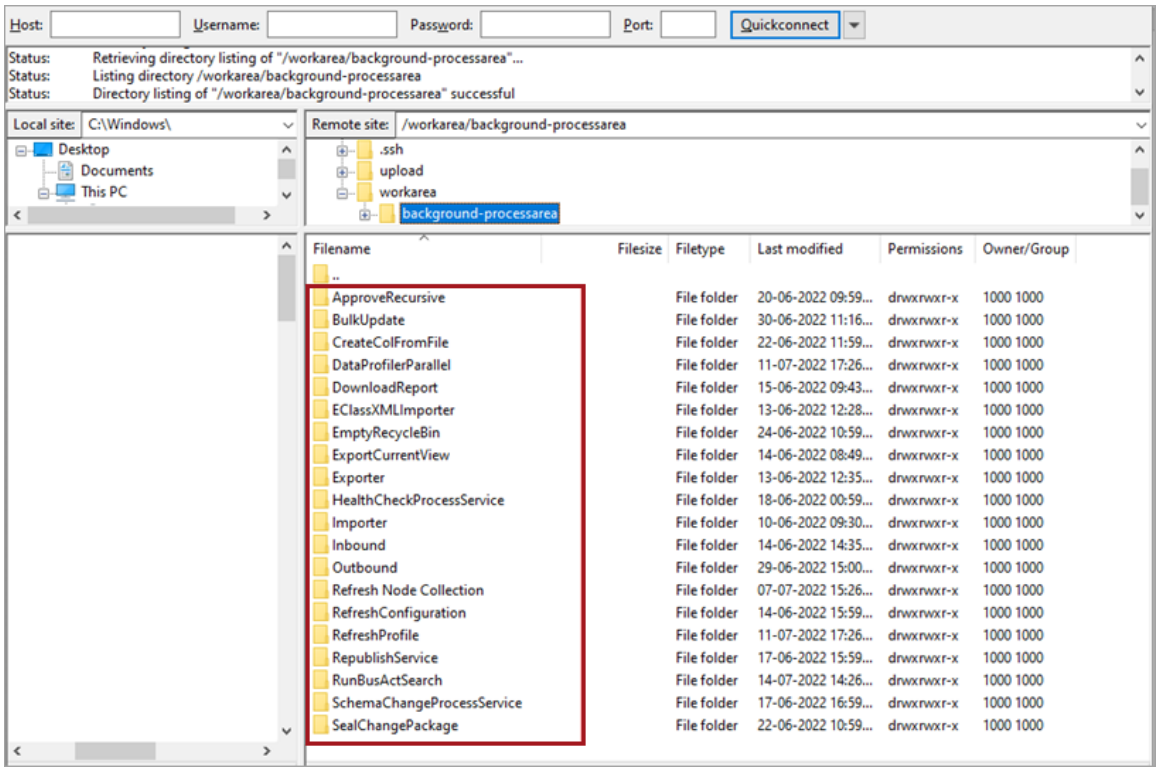
Background Process Auto Delete Settings

Automatic deletion of background processes is also managed through a configuration. The configuration defines the number of hours after the background processes ends that it will be deleted and is based on the specified background process template.

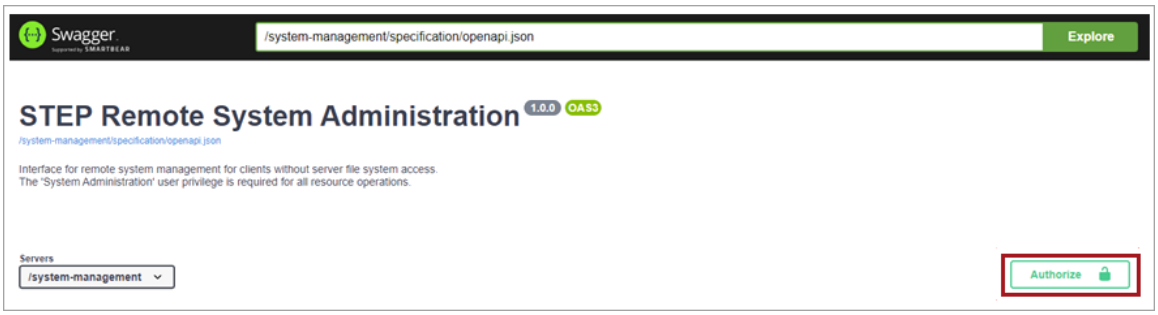
1. Set up an SFTP user in the STEP SaaS Self-Service UI, if you have not done so already, and pay attention to the SFTP URL. It will typically be: `<environment-name>-sftp.mdm.stibosystems.com`.
2. Use your preferred SFTP client application to connect via the SSH key associated with the previously created SFTP user.
3. When connected, you will have folders for 'upload' and 'workarea' (as shown below):



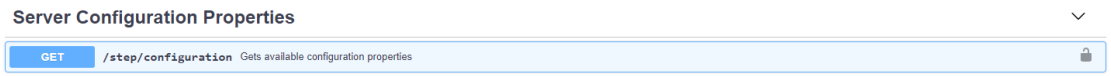
4. List the subfolder names in the 'background-processarea' under 'workarea' to view the available process templates.



5. Configure the auto-delete hours of the background process templates by using the System Management Swagger UI as follows:
   - Access the System Management Swagger UI, typically at the URL: `https://<system name>.mdm.stibosystems.com/system-management/swagger-ui`
   - Authenticate yourself with the 'Authorize' button using the credentials of a user in STEP with administrative privileges:

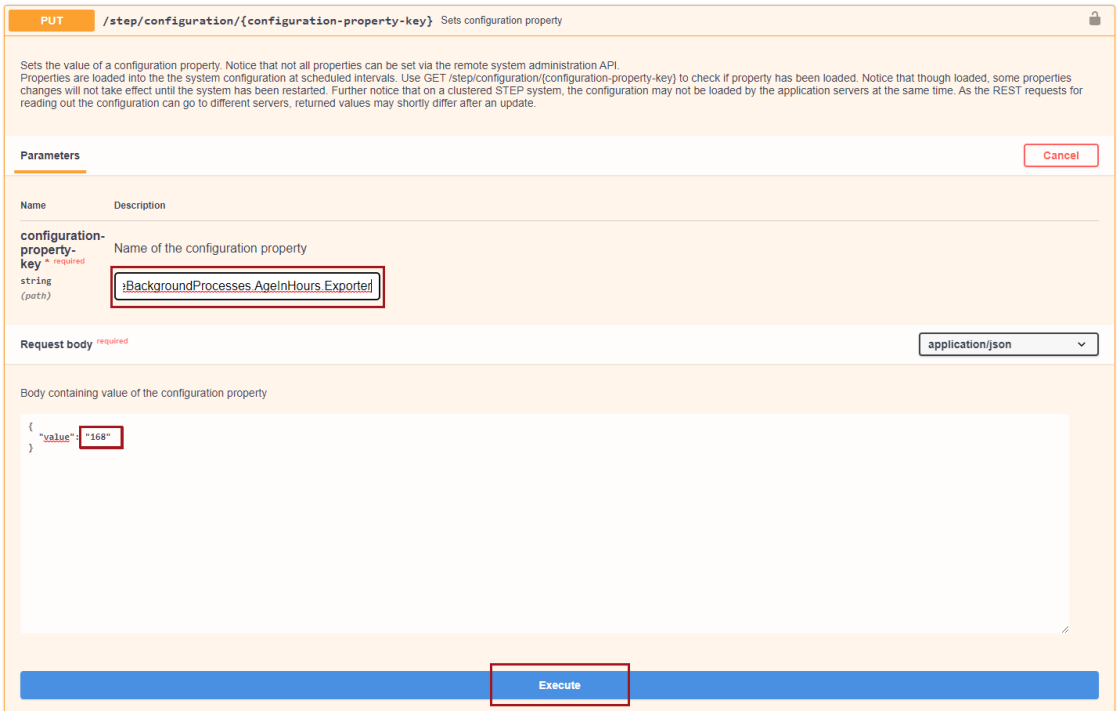


   - Check to determine if any of the case-sensitive parameter(s) have been set already using the GET /step/configuration endpoint:



   - Add or edit the templates to define the number of hours that should elapse prior to the automatic deletion of the template's background process.

     For example, the most important templates are generally Exporter, Importer, Outbound, Inbound, and Web Publisher. The parameter for Exporter can be configured to auto delete after 168 hours as follows:


PUT
/step/configuration/{configuration-property-key}
Sets configuration property
Sets the value of a configuration property. Notice that not all properties can be set via the remote system administration API.
Properties are loaded into the the system configuration at scheduled intervals. Use GET /step/configuration/{configuration-property-key} to check if property has been loaded. Notice that though loaded, some properties changes will not take effect until the system has been restarted. Further notice that on a clustered STEP system, the configuration may not be loaded by the application servers at the same time. As the REST requests for reading out the configuration can go to different servers, returned values may shortly differ after an update.
Parameters
Cancel
Name
Description
configuration-property-key * required
string
(path)
Name of the configuration property
BackgroundProcesses.AgeInHours.Exporter
Request body required
application/json
Body containing value of the configuration property
{
"value": "168"
}
Execute

Web UI Configuration Recommendations

This is one of the data gathering methodologies and recommendations for functional performance improvement. The full list is defined in the **Performance Recommendations** topic.

The Web UI Designer is flexible and can configure a Web UI in many ways. However, not all configurations perform equally well. Therefore, it is important to consider performance when configuring a Web UI. A screen with little data and limited functionality will load faster than a screen with lots of data and functionality.

For more information, refer to the **Healthcheck** section of the **Administration Portal** documentation.

The following actions can limit the performance impacts while using Web UI:

- Web UI Component Report
- Use multiple Web UIs
- Use small dedicated Web UI screens
- Avoid using images in multi-select Web UI screens
- Use 'lazy' loading for Web UI screens
- Use type ahead for LOVs in Web UI screens
- Correctly configure status selectors in Web UI
- Privileges in Web UI configuration
- Consider In-Memory for Web UI screens

Web UI Component Report

The Web UI Component Report provides an in-depth look at the components configured for all Web UIs. Of particular interest are those in the end-stages of their lifecycle. Use this report to replace outdated components and update configurations improve the upgrade process. The reported components are those that have been:

- Deleted entirely: will not render
- Withdrawn: once removed cannot be added again
- Superseded: will work as usual, but a new and improved component is available

The Web UI Component Report link is displayed at the top-right of the system Start Page and in the popup that displays when accessing the Web UI designer. You can also access it using the URL http://[ENVIRONMENT]/webui/componentreport (modified with your environment name).

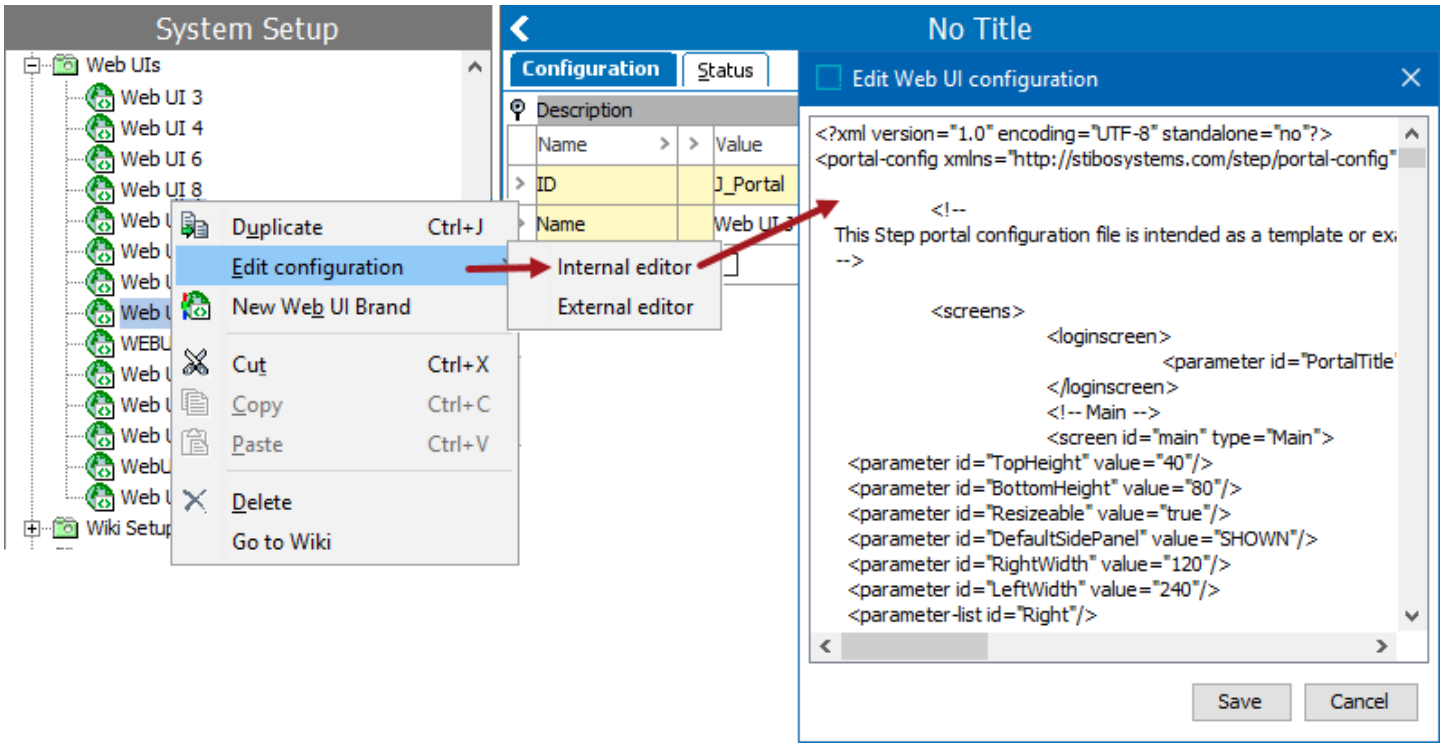
For more information, refer to the **Web UI Component Report** section of the **Web UI Component Basics** topic in the **Web User Interfaces** documentation.

Recommendation

For optimal Web UI performance, remove the superseded, withdrawn, and deleted Web UI components or replace with the newer available Web UI components.

Use Multiple Web UIs

Web UI configurations are stored in System Setup under the Web UIs node. When a user logs into the Web UI and opens the homepage, the corresponding configuration XML file is loaded. If the Web UI configuration XML file is large, then the Web UI homepage takes additional time to load. Edit the Web UI configuration to view the XML, as shown below.



For more information, refer to the **Managing Web UI Configurations** topic in the **Web User Interfaces** documentation.

Recommendation

When the Web UI configuration XML is more than 25,000 lines, split the configuration into separate Web UIs where each is used for a specific purpose.

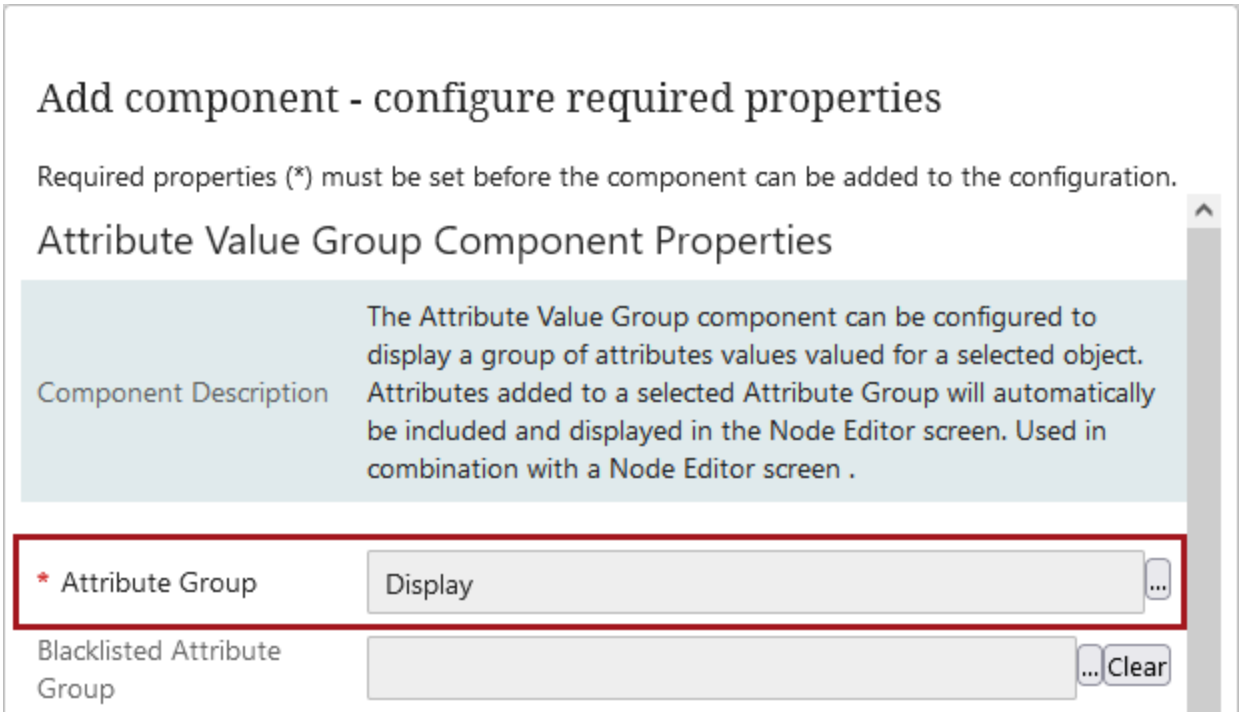
Use Small Dedicated Web UI Screens

A typical Web UI screen fetches all attribute values from the attribute group defined in the Web UI screen. The screen then filters out the attributes based on the validity of the product type and user privileges.

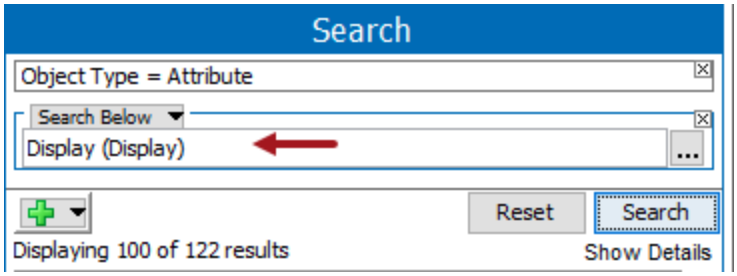
Configuring attribute groups for the Web UI screen which contain a large number of attributes can have a negative impact on the loading time of the screen because it involves:

- fetching all attributes from the attribute groups, and then
- filtering out attributes for display based on the validity of the product type and user privileges.

To reduce the screen load time, first determine which attribute groups are used in the Web UI screen, as shown below.



Then analyze how many attributes each attribute group contains, using the Search Below option.



Recommendation

Avoid using attribute groups with more than 100 attributes. When configured attribute groups contain more than 100 attributes, consider creating attribute groups specifically for Web UI display. Organize the attributes to be displayed on the Web UI screens into these specific attribute groups.

Avoid Using Images in Multi-Select Web UI Screens

A multi-select screen displays a selection of items in a table. The table view also allows for thumbnails of the items to be displayed. However, loading many items with their thumbnails naturally requires fetching these thumbnails from the file system or database, which may have a negative impact on the load time of the screen.

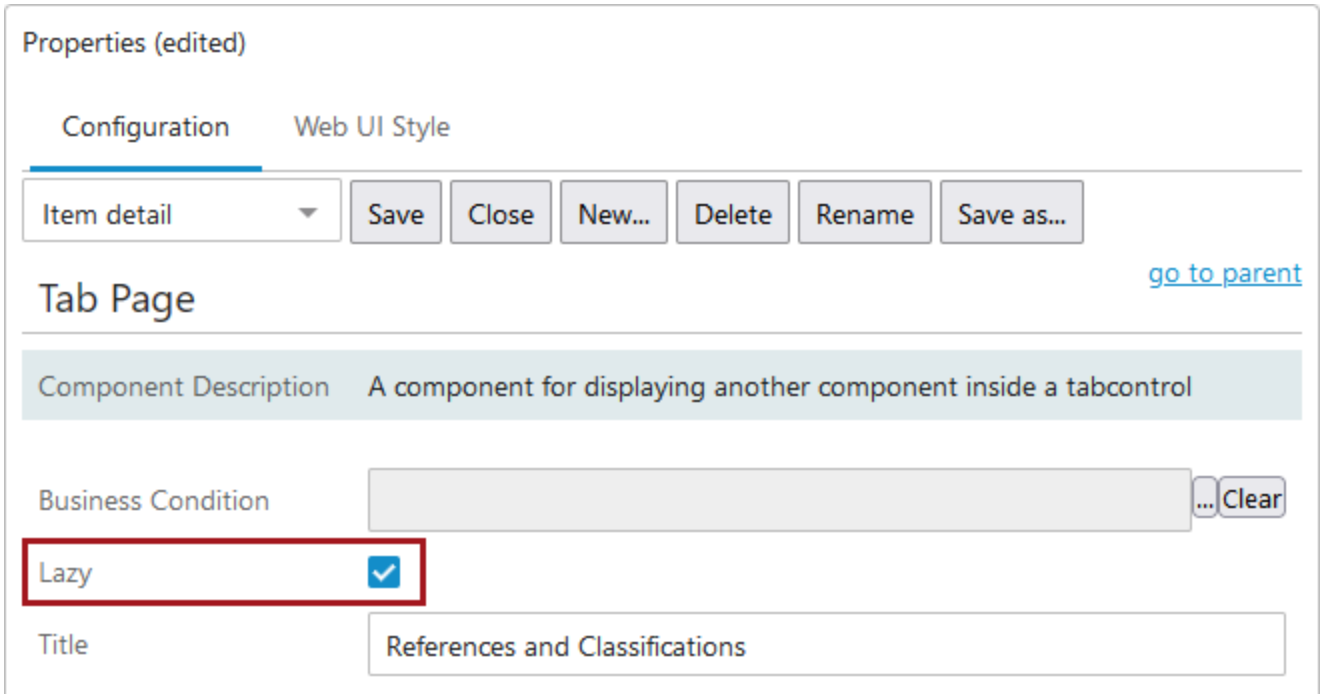
Recommendation

Reconsider displaying thumbnails (or no images at all) in multi-select screens when many items are displayed and load time of the screen is considered slow.

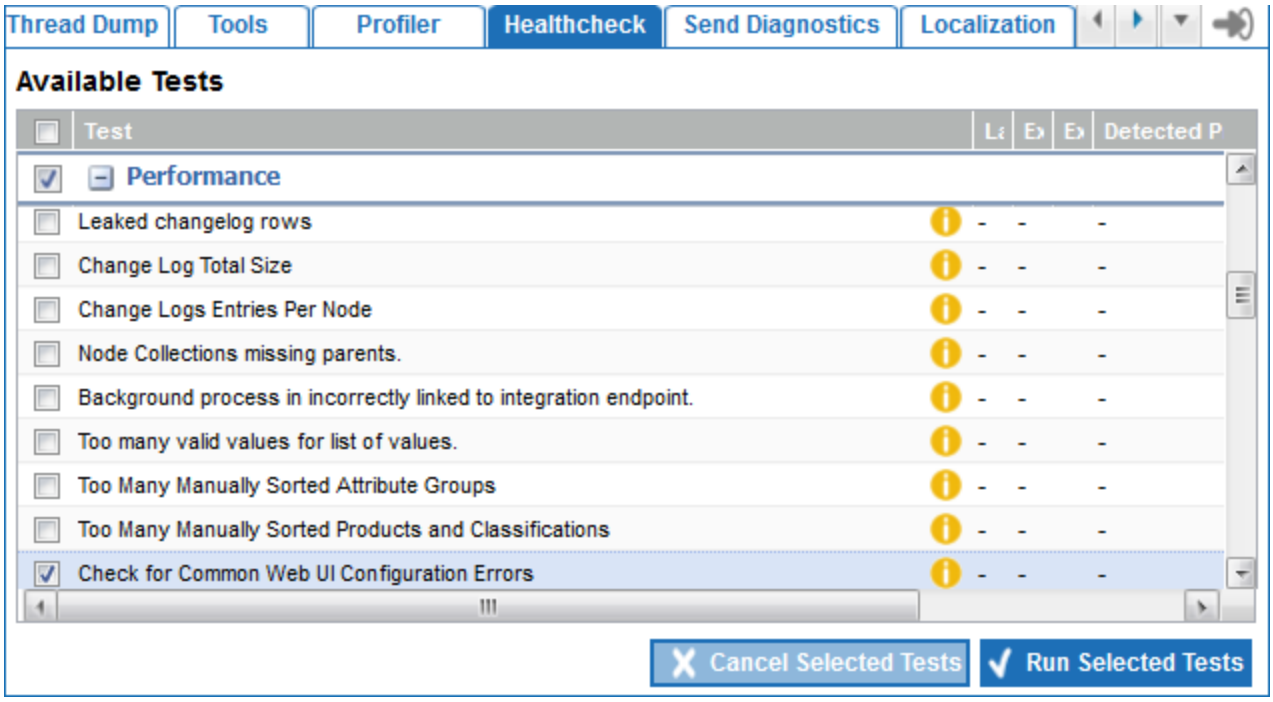
Use 'Lazy' Loading for Web UI Screens

For Web UI tasks where the user needs to inspect many aspects of an item, the recommendation is to split the information out into multiple tabs. However, by default, the Web UI loads the screen including all tabs, which may result in a slow loading time for the Web UI screen.

In Web UI designer, the 'Lazy' parameter allows you to apply 'lazy loading' on these Tab Page screens. When enabled, components are 'lazy loaded' and rendered only when a tab is displayed, which can reduce screen load time.



Use the Performance section of the STEP Health Check in the Admin Portal to analyze which Web UI screens can be lazy loaded by running the 'Check for Common Web UI Configuration Errors' healthcheck shown below.



For more information, refer to the **Healthcheck** topic in the **Administration Portal** documentation.

Recommendation

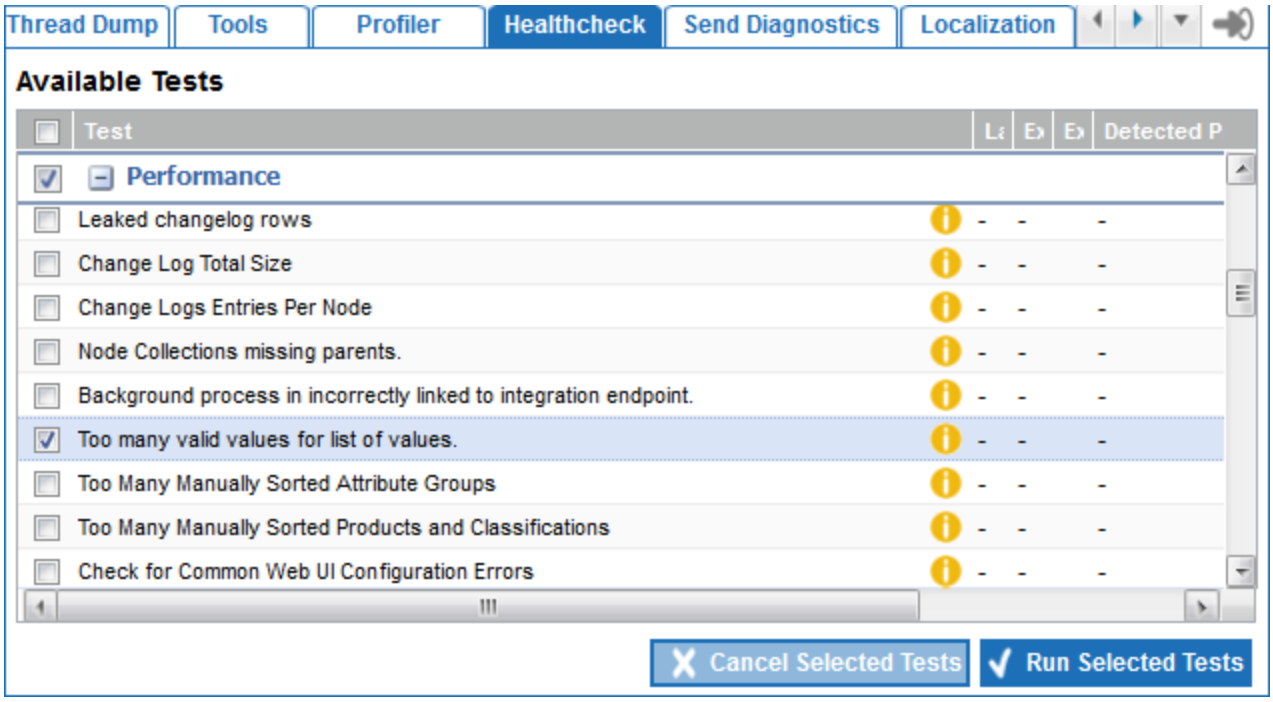
When possible, lazy load data on tabs.

Use Type Ahead for LOVs in Web UI Screens

Web UI screens using attributes with a large list of values (LOVs), the load time of the Web UI screen can become slower. The following LOV type-ahead sharedconfig.properties are available to optimize the Web UI screen load time:

- **Portal.ValueGroup.LOV.ForceTypeahead** forces type-ahead for all LOVs in the Web UI. Set this property when there are attributes used in Web UI screens and many of the LOVs used are large.
- **Portal.ValueGroup.LOV.ForceTypeahead.Exclude** forces type-ahead for certain LOVs in the Web UI. Certain specific LOVs can be configured not to be forced to user type-ahead functionality. Use this for the LOVs with small number of values and for the LOVs which are required to not be typed-ahead.

Analyze which LOVs have more than 5,000 values by running the 'Too many valid values for list of values' healthcheck. Also review Web UI screens to find where large LOVs with fewer than 5,000 values are being used.



For more information, refer to the **Healthcheck** topic in the **Administration Portal** documentation.

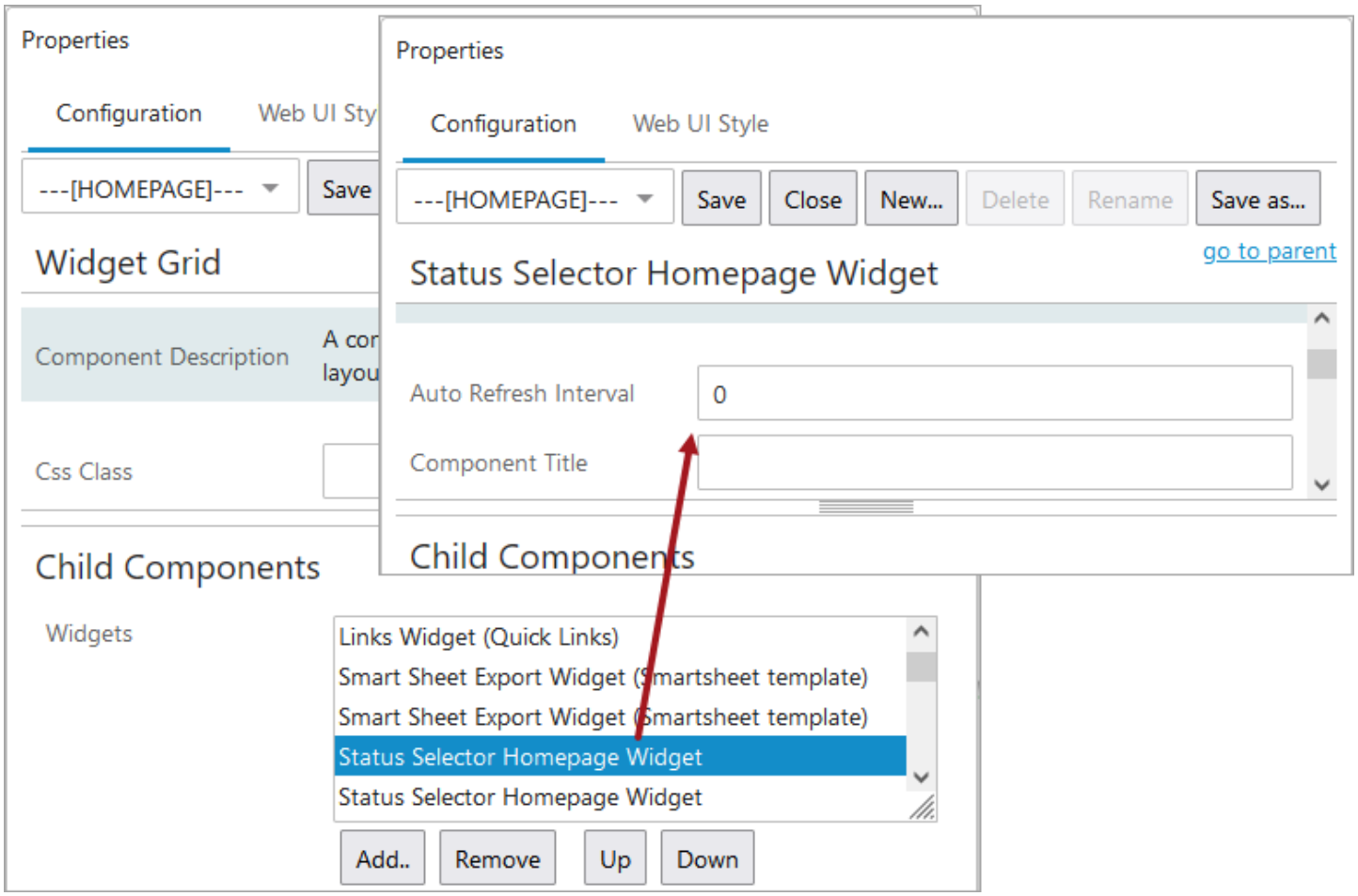
Recommendation

Set up typeahead for large LOVs.

Correctly Configure Status Selectors in Web UI

The workflow Status Selectors Homepage Widget is configured to poll for updates in the background. The statuses are updated whenever the user submits an item. However, if there are many status selectors and/or the update polling is set very fast, the Web UI Homepage screen may suffer from poor performance.

In design mode, double-click the Status Selector Homepage Widget to display the 'Auto Refresh Interval' parameter. The recommended value is 60 seconds. Disable the refresh interval by setting the value to 0.



For more information, refer to the **Status Selector Homepage Widget** topic in the **Web User Interfaces** documentation.

Recommendations

- Limit the number of status selectors, generally no more than 25 status selectors on the Homepage screen.

  Download the Web UI XML to review status selectors used in a Web UI and search for:

  - **StatusSelectorHomepageWidget** to find the status selectors on the homepage.
  - **StatusSelectorWidget** to find the status selector on another page.

- Do not set the Status Selector Homepage Widget 'Auto Refresh Interval' parameter to less than 60 seconds.

  Download the Web UI XML and search for **AutoRefreshInterval** to find auto refresh intervals on status selectors.

- Be aware that including many nodes in a workflow being calculated in the Status Selector may increase loading times.
- Reduce load times. On the Web UI designer for the Status Selector Homepage Widget properties screen, Advanced section, check the 'Use Content Indicator' parameter. This shows a button to manually load the count, instead of displaying the exact count automatically.

  Download the Web UI XML and search for **UseContentIndicator** to find the content indicator used on status selectors.

Privileges in Web UI configuration

Although privilege restrictions can be set in the Web UI configuration itself, excessive privilege checking in the Web UI XML configuration can degrade performance.

Recommendations

Download the Web UI XML and search for **restrict=** and analyze if these user restrictions are necessary.

Consider In-Memory for Web UI Screens

In-Memory can improve performance of the Web UI screens. In-Memory makes the Web UI more responsive in general, but especially when the Web UI screens include multi-object display, complex data models, and/or data with a long history of writes.

For more information, refer to the **In-Memory Database Component for STEP** section of the **Resource Materials** section of online help.