OBJECT MIGRATION
BEST PRACTICES

Table of Contents

1. WHAT IS OBJECT MIGRATION?

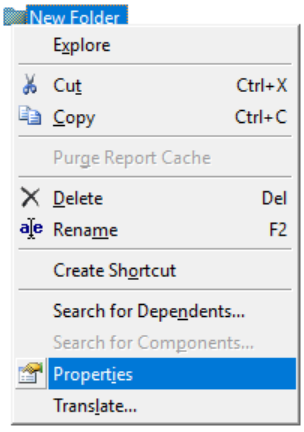
Object Migration is the process to migrate or move objects from one environment to another environment. For example, the objects developed in a Development environment will be copied to a UAT environment to test and validate the reports/dossiers.

What are the benefits of having a streamlined migration process?

- Reduced uncertainty and risk
- Reduced delays, wasted time and unexpected costs
- Improved corporate performance
- Maximized ROI of Enterprise applications
- Efficient and effective business processes
- A measurable and accurate view of data
- Better accountability and ability to meet compliance targets

2. OBJECT PROPERTIES

Reviewing and Managing Properties of MicroStrategy Objects may be a task for Developers and Architects when building and Application, but it is one of the main responsibilities of a Platform Administrator when synchronizing objects across environments. It is important to have good understanding of Object Properties, before attempting to manage Object Migrations:

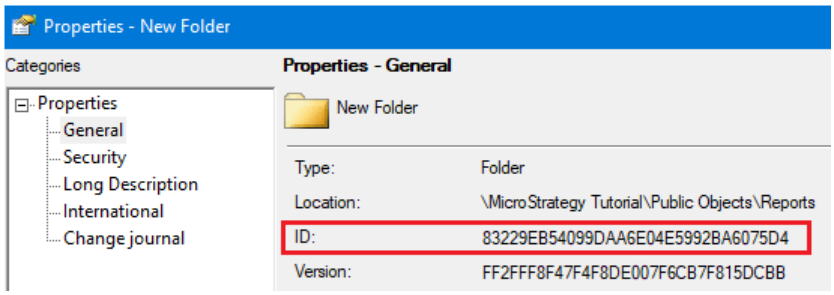


OBJECT IDENTIFICATION

Much like Records in the Database can be held distinct by using an identifying or versioning information, an object in MicroStrategy also needs to be identified, referenced and kept track of.

GUID

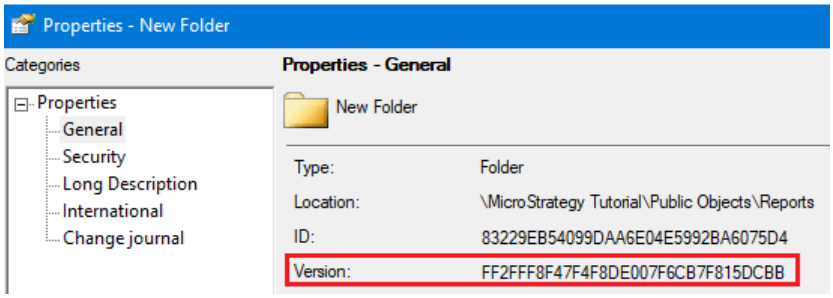
Objects in MicroStrategy get referenced by their GUID (Globally Unique Identified). This helps to maintain dependencies while object definition, naming or other properties get modified. If two objects have the same GUID, they refer to the same object. When synchronizing an application between environments, the complete set of objects being a part of that application need to be updated accordingly. IDs will therefore play a key role when searching dependencies to keep the entire groups of objects synchronized so that entire application can be migrated properly.



Version

Objects in MicroStrategy will also have a Version ID which changes every time the object is modified. This will play a key role when attempting to find out if object at its latest stage has already been synchronized across environments or if it exists in an obsolete state somewhere.

When in doubt about a single object, Platform Administrator can check the Version ID manually in Object Properties – and if it's a match, it means its definition will also be matching. However, it's important to note that definition of component objects or default project settings may still differ if object with same Version ID is behaving differently.



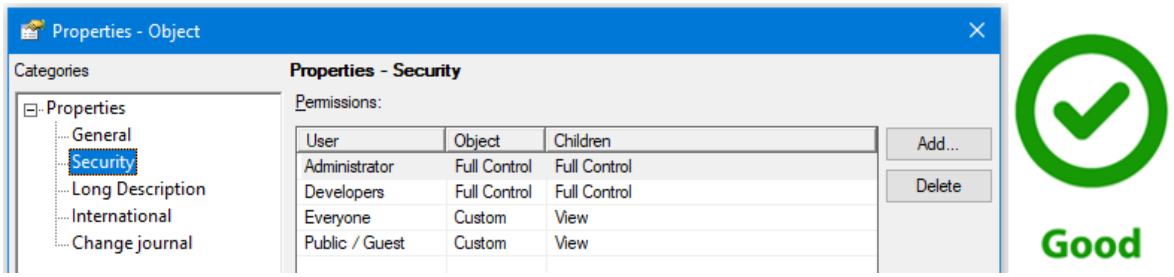
ACCESS CONTROL LIST

MicroStrategy Objects get assigned Permissions in conjunction with Users or User Groups, just like objects in an Operating Systems ("Security" in Windows or "row" in Unix and Linux systems). One of the main responsibilities of the Platform Administrator is to ensure that, after objects get synchronized, all End Users will have access to objects they are supposed to see and will be declined access to objects they must not see.

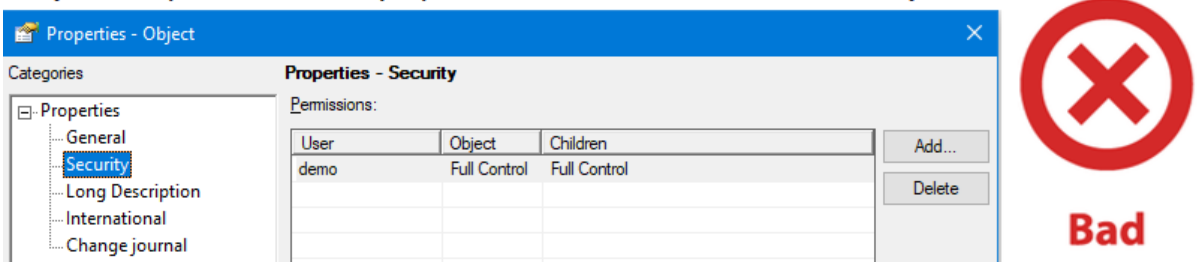
Assigning Access

MicroStrategy Objects will inherit their Access Control Lists from the Folder they are created in. When object is moved to another folder, it will maintain their ACL, however, when the object is copied, the ACL will be assigned from the target folder, just like when creating the new object.

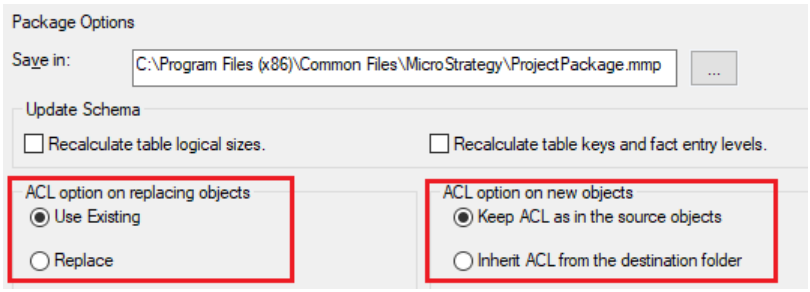
Developers and Architects must be aware of those settings to successfully build applications as a team. In some cases, Platform Administrator might be asked to help with modifying the Access Control List, as his/hers credentials allows to bypass the ACL settings. Example of a common scenario is shown on below images:



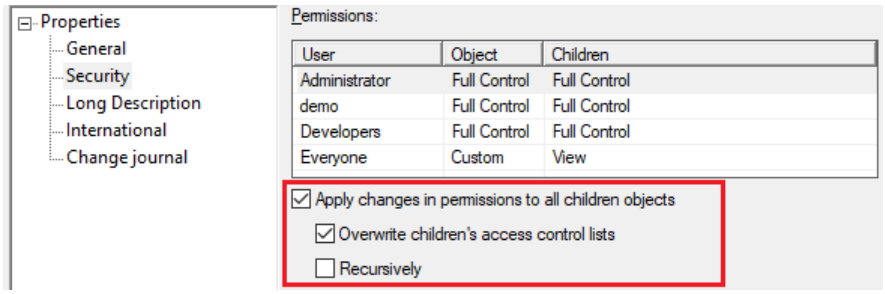
Object developed in someones "My Reports" folder and later moved under "Public Objects"



When migrating objects, it's possible to choose if ACL should be kept from the object in source environment or assigned from the object or (if new object) from folder in target environment.

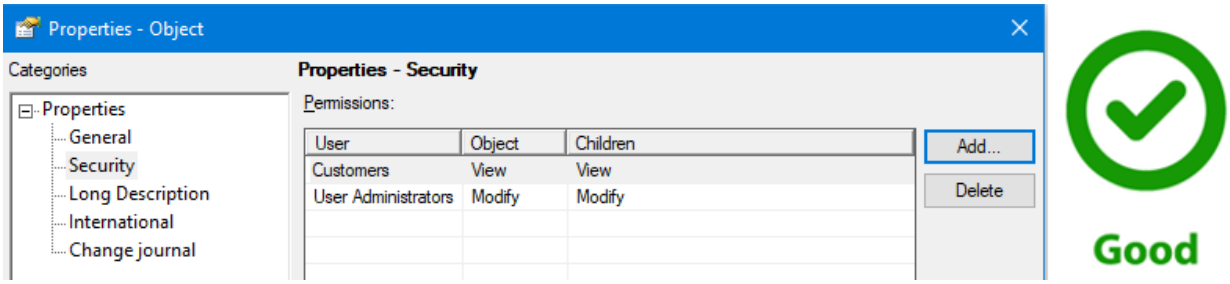


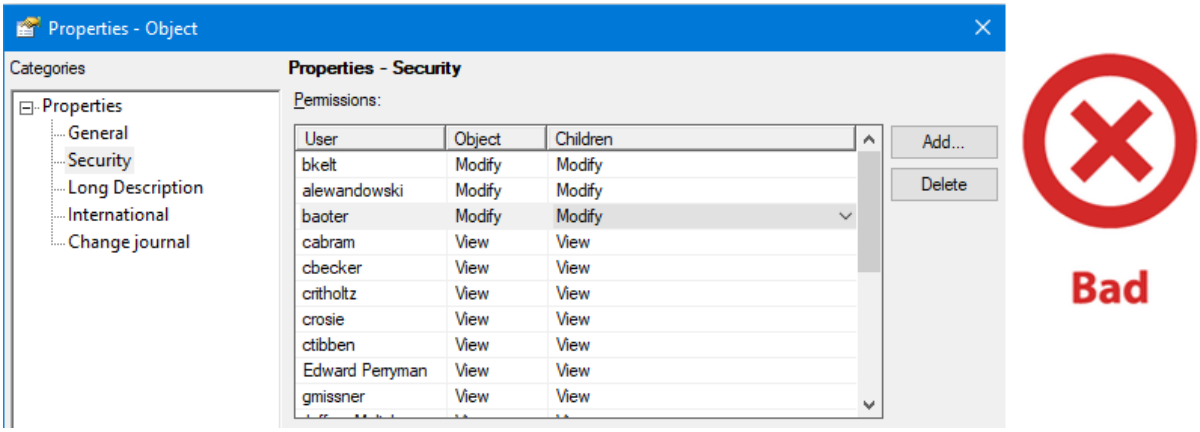
If all objects under given folder need to have a specific Access Control List assigned, then Platform Administrator can also overwrite access of all objects in that folder after migration with option to overwrite objects in sub-folders as well (recursion):



User Groups

Users and Users Groups are configuration objects and while Users may be added or removed all the time, User Groups usually do not change that often. The reson for this is to make it more efficient for Platform Administrators to assign user permissions to objects. It is a good practice to keep only User Groups in the Access Control List of all Folders and Objects underneath them:





It is therefore another good practice to keep User Groups consistent between environments to allow assigning them permissions to objects as early in the migration process as possible:

Those Groups can be empty in some of the environments (for example, Developers reside only in the Development environment, they rarely need to be in Production environment). It's important that, in the environment where given Group contains Users, they will have appropriately assigned permissions shortly after each object migration is executed. When a new user is added, simply assign him/her to a group.

3. MIGRATION PROCESS

Object Migration is a repeatable process and as such, it should be designed to be most efficient and followed by all involved resources to ensure simple and risk-free object synchronization.

ENVIRONMENTS

Depending on the purpose, the Analytics Platform deployment will be divided into multiple environments – at least as separate instances of the Intelligence Server with dedicated metadata.

Those environments will contain same MicroStrategy Projects at various stages of application evolution – until synchronized. Reason behind that is separating environments where objects need to be Build and Reviewed from the environment where objects will only be Consumed.

Standard Set

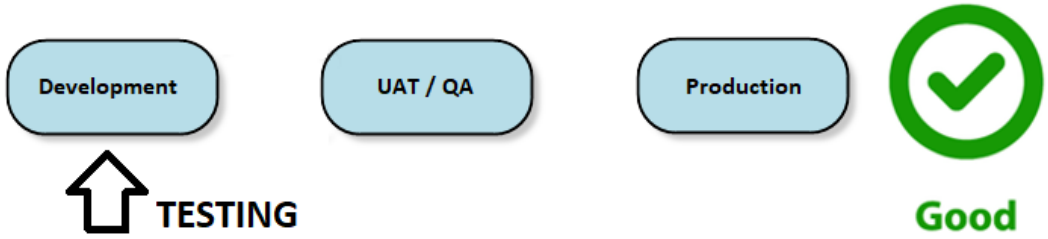
Most common Set of Environments is:

- **Development** – where the Applications are created
- **UAT / QA** – where Users are testing the Applications
- **Production** – where Applications are consumed by target audience

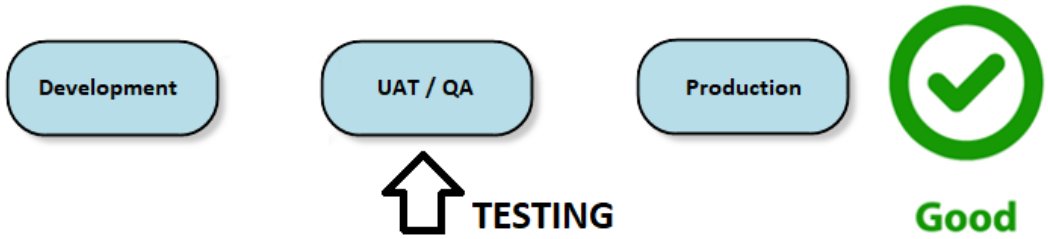
Very often applications will have their stages of development or with time will require further enhancements. Building or modifying an application often requires multiple steps and checks before it can be made available to the End Users. For this reason, it is never a good idea to perform development of a shared content in Production environment.

Where to Test

While Application is being developed, it is most efficient if Developers are empowered to understand the application purpose and test if the newly created set of objects are fulfilling the requirements before requesting an object migration.

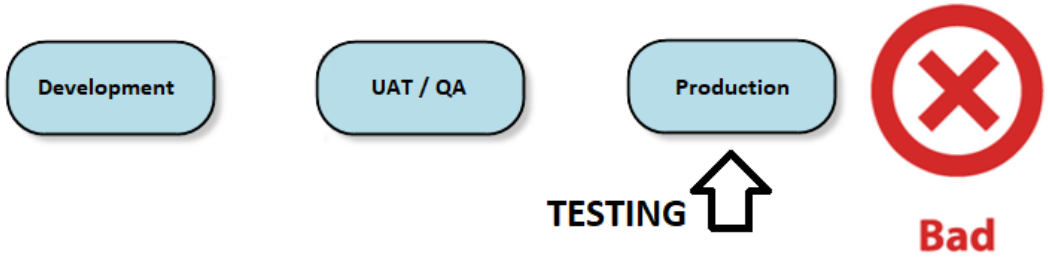


When Application development is finished, the related Objects will need to be migrated from Development environment to UAT / QA environment, where users can proceed with their tests of all the common use cases, ensuring needed functionalities and reliability of the application KPIs:



Once Objects reach Production environment, the application has been made available to the final audience and any flaws not caught in the UAT/QA environment may result in making business decisions based on incorrect information.

If for any reason Users prefer to wait until the Application reaches Production Environment, to see if it fulfills their requirements – it means the UAT/QA environment has not been set up correctly. Often Data Warehouse (WH) is too obsolete to support testing – and therefore it is a good practice to keep the latest snapshot of Production WH available for Application testing. In some cases, it may even be necessary to connect UAT/QA to Production Data Warehouse, but it is never a good practice to perform testing in Production environment:



After many iterations of testing and several trial and errors, it may be tempting for Developers to try and use UAT/QA environment for applying quick fixes to the Application. This is not a good practice, as in this case synchronization of Development and UAT/QA environments would require reversing the object migration process.

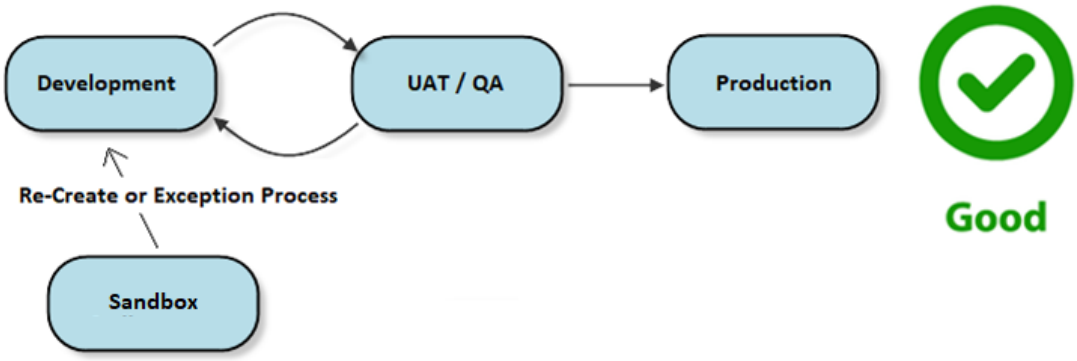
Development is separated for a reason, which is: clean and stable environment designated for User Acceptance Testing / Quality Assurance. Developers wanting to build objects in UAT/QA usually originates from inability to perform thorough testing of their work due to environment or Data Warehouse limitations. While in some cases Developers are not supposed to be exposed to the real production data, they still need to be exposed to all the Platform infrastructure, variety and velocity of the data that applications will eventually use, in order to perform realistic functional testing.

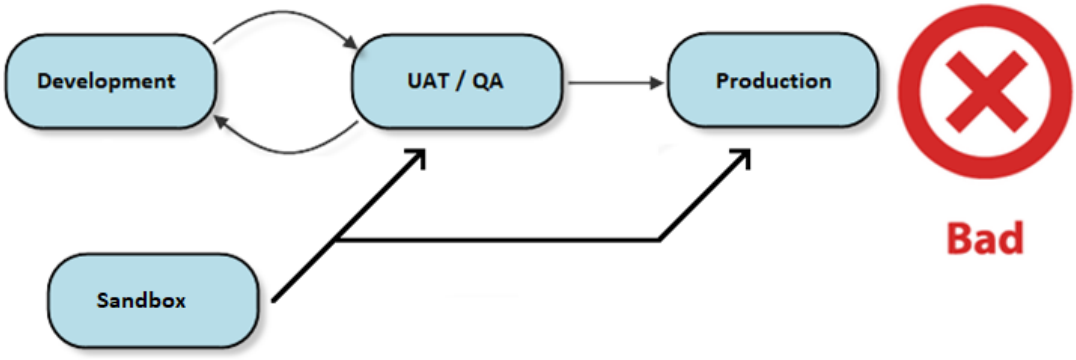
Sandboxing (optional)

As implied in the previous section, the Development is not the most stable environment, which is why Platform Administrator needs to take all the precautions when performing Object Migrations to environments that are supposed to be stable.

Some tests of Global Changes and Proof of Concept work could impact the development flow so much that could create confusion, slow down or jeopardize object creation and enhancement process. In some cases, it is better to keep those experiments separate from Development environment; such environment is usually called “Sandbox”.

This environment should not be in the Migration Loop at all, unless the POC developed there is very large and valuable, and it wouldn’t make sense to re-create it by hand. In such case a single object migration could be performed, as an exception, from Sandbox to Development:



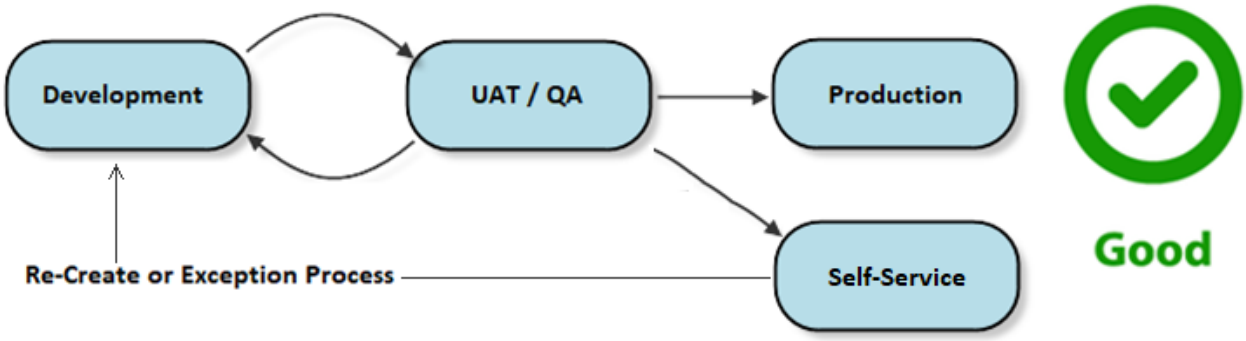


Self-Service (optional)

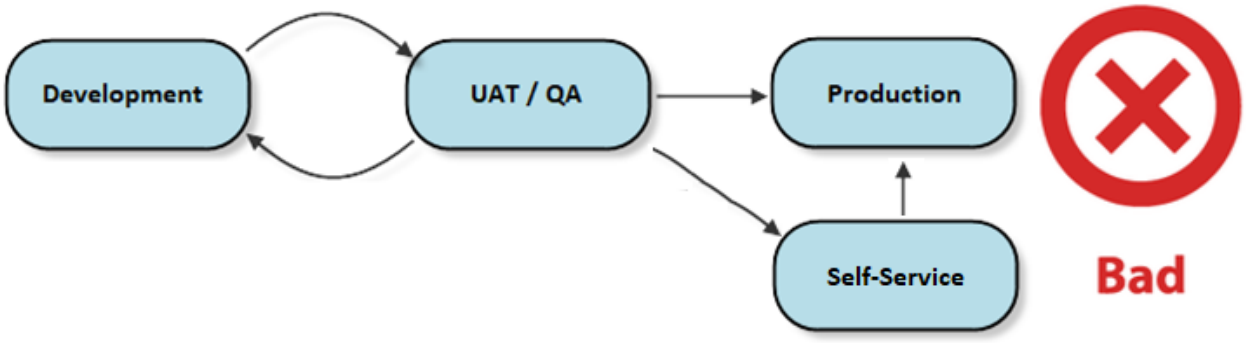
Production environment may be used in many ways, which sometimes is a good reason to split the environment. For example: Environment 1 could contain simple reports, that need to be available promptly to a very high volume of users every minute of the day. Environment 2 would store a very large Dashboards that recalculate long term trends any time there is a major change in the market factors – and those numbers would be only visible to executives. Keeping those together would obviously affect each other's performance at random times.

One of more common scenarios is separating the Self-Service environment where End Users can execute Ad-Hoc requests by resolving a predefined set of prompts or build their own Dossiers on a predefined set of Cubes, ensuring they will always get correct data whatever they construct.

Eventually, users might like to share their work to a wider audience as a part of an Application. The best practice in this case is to plan development using the Self-Service work as requirements exception. Alternatively, it's ok to migrate the object to Development, apply all necessary enhancements and proceed with standard Object Migration procedure from there.



This way the process is preserved, and all objects get tested in UAT/QA before final deployment.



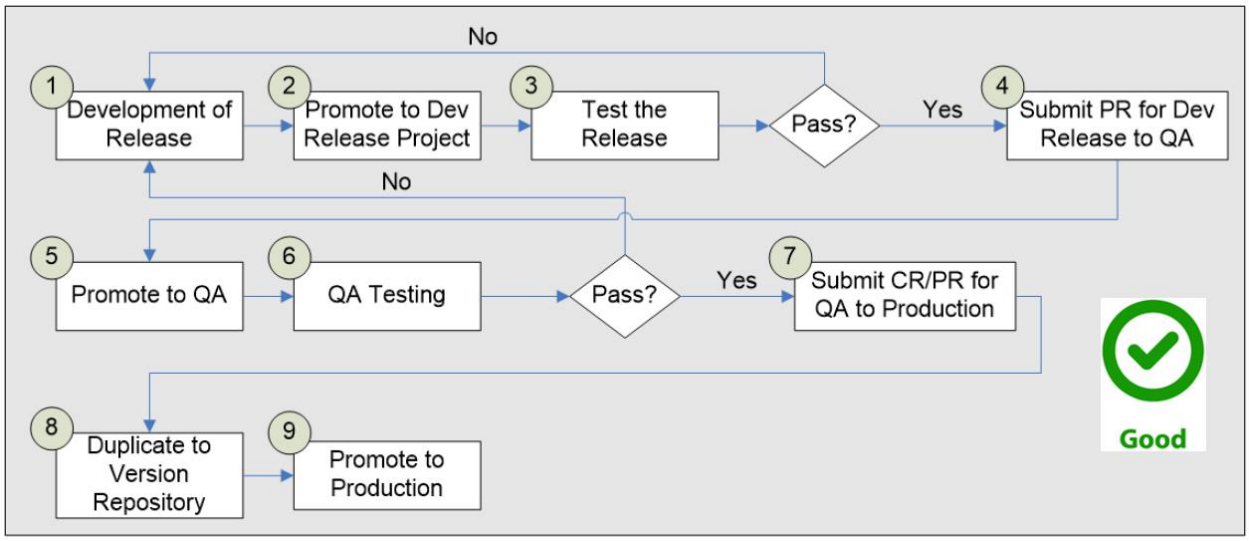
WORKFLOW

The workflow of Object Migration should be simple and time-efficient but must also consider all possible scenarios and respect dependency on any internal corporate standards, in case they are applicable and related.

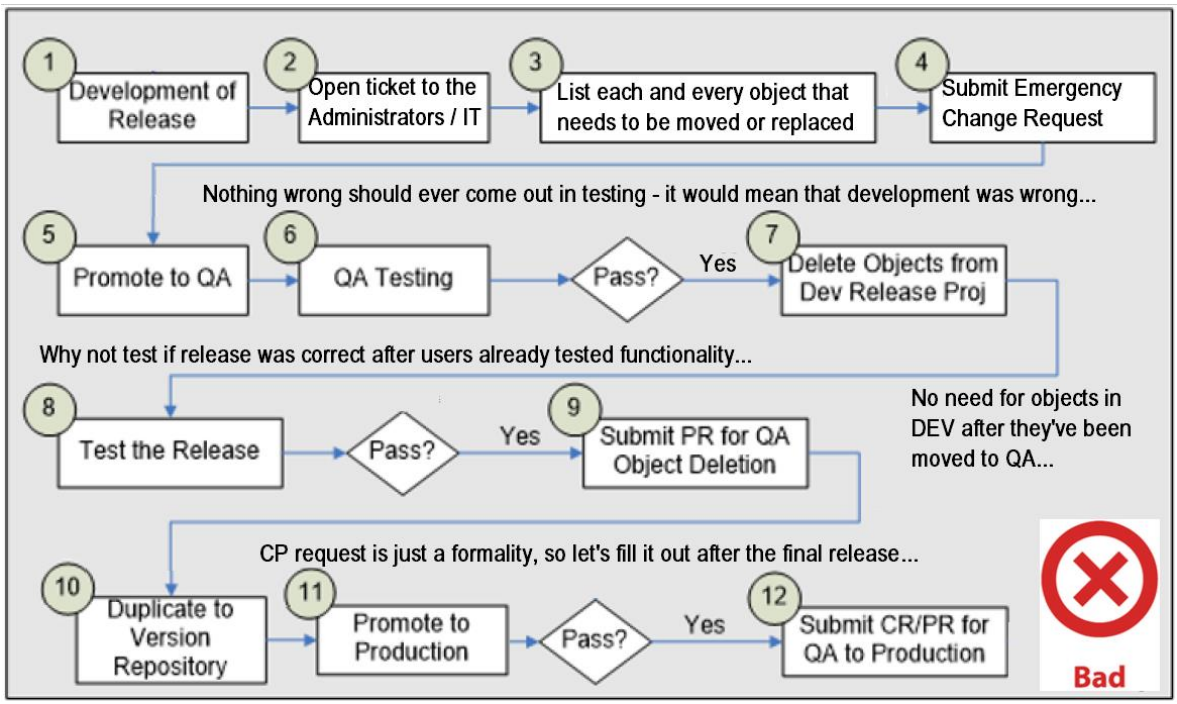
Change Control

Oftentimes, it may be tempting to completely re-use a "Change Process" from ETL, Data Warehousing or IT team and force to re-use an existing ticketing system designated for code development (not Object Migrations). Unfortunately, saving a little bit of time during planning phase, may result in great loss of time in day-to-day work and eventually in users not respecting all steps of the process – looking for shortcuts and cutting corners whenever possible.

Below is an example of a good migration process where Promotion Request (PR) is easy to fill out by Developers. Moving Application to Production requires additional Change Request (CR), but it is generated automatically, and approval is needed only from one User/Requestor who represents the testing team of a given Application.



Below is an example of a process which got so complicated, that no one would follow all the rules or try to imagine mapping its steps into a flow chart. However, if at some point someone would take and honest look at the what people are doing and map all the steps that really happen during each release, it would probably look something like this:

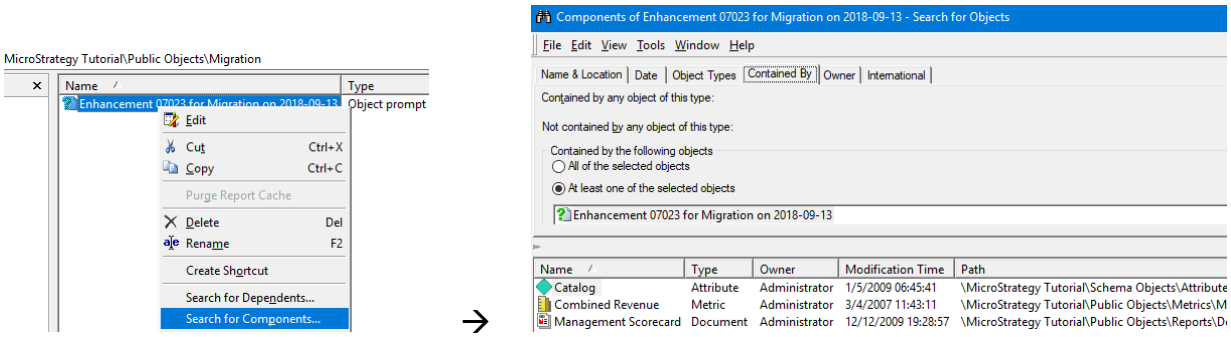


Migration Ownership

Another thing worth avoiding when planning a migration process is dispersal of responsibility. It's a common bad practice to give more responsibility to Developers, since they know objects they have created and therefore give them access to do Object Migrations or prepare packages. While this might work in a perfect world, at the end of the day it is the Platform Administrator who takes responsibility for every part of the release process.

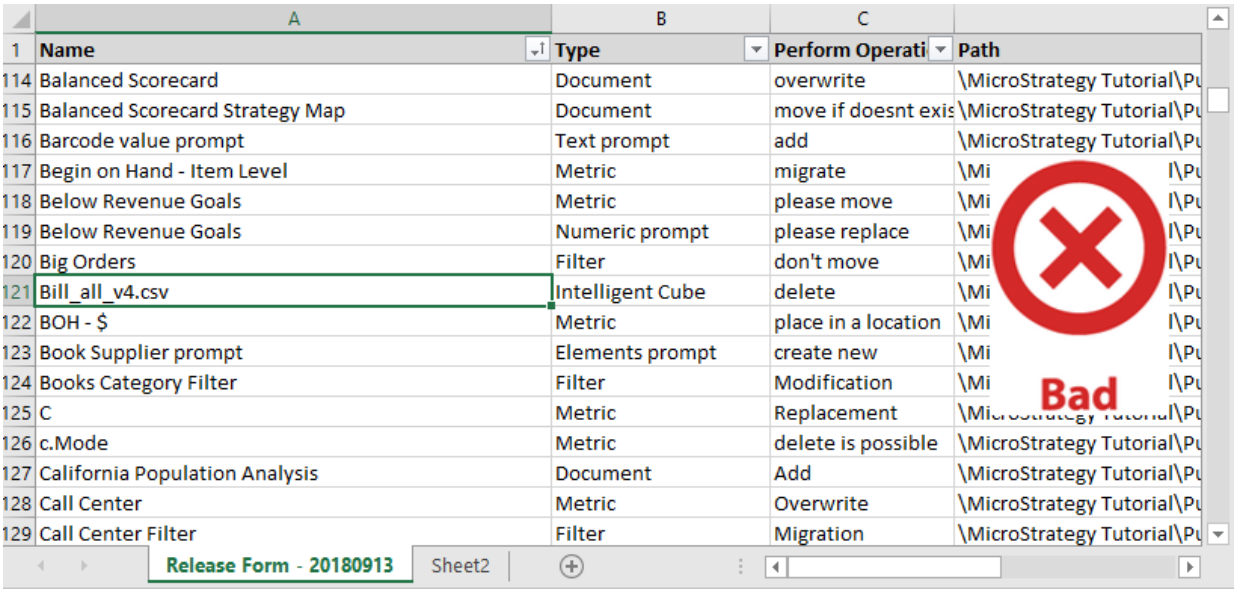
At the same time, Platform Administrator will need to rely on Developers and Architects to provide him/her with the necessary information to migrate appropriate objects between environments.

One of the ways to meet halfway is to ask Developers to prepare Object Prompts that would contain all Migration-Ready Objects as components:



This approach may have some downsides. There's still room for human error and it could be said that Developers are really owning the Object Migration process if Platform Administrator gains no understanding of what part of which application he's moving with his object migrations.

Another customary practice is using release forms. However, listing every single object that need to be moved along with specifying which to modify or move as new object can be very tedious. It's time consuming for Developers to create such list and for Platform Administrator it's just not efficient to move objects one by one:



Instead, it's better to take advantage of object relationships and folder structure. Developers could then make use of more descriptive language, like in below examples:

- "Please migrate all Components of dashboard <dashboard_name>"
- "Please synchronize all objects Inside folder <path>/<folder_name>"
- "Please move all Dependent schema objects of logical table <table_name>"

Those are simple instructions that Platform Administrator can consciously follow when using Object Manager (more about this administrative tool mentioned in Chapter 3).

Release Testing

After the release to UAT/QA environment, designated testers will look at KPI calculations, functionalities and general feel of the Application as a whole. Before providing them with a signal to "go ahead", it's a good practice to check if the release was successful and if objects behave the same way in the Target (UAT/QA) as in the Source (Development) environment.

This step will also remind to Update Schema or Re-Execute Cubes that have been modified in UAT/QA environment with a correctly performed migration. It is generally a good practice to investigate the situation if suspecting any human error rather than use any drastic measures:

If Version ID of a dashboard or dossier is matching but it behaves differently between environments, it could be one of the underlying datasets (Cube or Grid Report) - possibly one of its components hasn't been included in the synchronization. It could also be related to database or global setting.

One of the simple ways to make sure that dataset along with all underlying component objects is in synch between environments is to ensure that it generates the same SQL in both environments. To perform this testing accurately for multiple datasets at the same time, it's beneficial to use Integrity Manager (more about this administrative tool mentioned in Chapter 4).

4. OBJECT MANAGER

Object Manager is a MicroStrategy administrative tool that allows Platform Administrators to perform Object Migrations between environments. It can be also used for reorganizing or copying objects within the same project, assigning Access Control Lists available in the same way as in MicroStrategy Developer.

Synchronizations (moving or replacing objects) in Object Manager are done between two environments at a time. It's possible to open multiple environments but a single move or replacement always happens between the two of them.

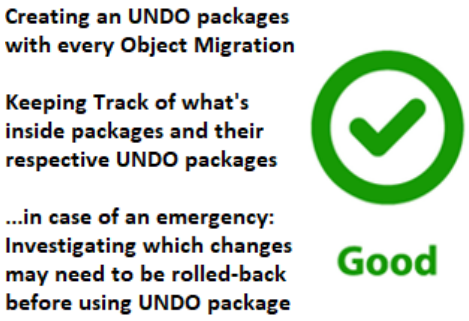
PACKAGES

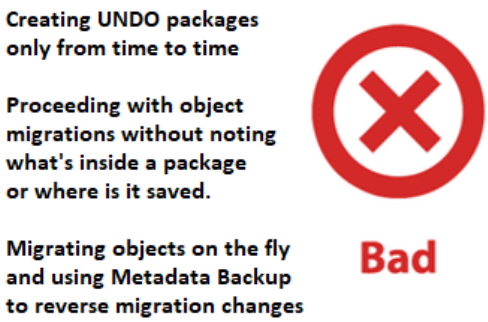
While it's possible to synchronize a selected group of objects between two environments on the fly, it is a best practice to use Packages. Those are a form of export of all planned changes based on the selected list of objects from the Source environment.

Create Backup

Using Packages allows the Platform Administrator to create a reverse package to undo some or all the changes performed with the synchronization. Creation of the "Undo Package" can only be done in the Target environment before importing the original package.

It's a good practice to create reverse packages for every single package that is about to be deployed as a form of backup. Using it is just like using a regular package and results in changing the objects back to the form they were in the Target environment before deploying the original package. It will also remove all new objects moved in the original package.





Grouping Objects

Common bad practice is trying to jam all objects needed to be moved or replaced by a given day into a single package. It is ok to perform a partial migration of schema and datasets before higher level application objects are finalized – it gives more time for testing the calculations before focusing on the look & feel, and it usually takes away time and complexity (and therefore effort and stress) from the future object migrations.

Following the development, it's also a good practice to group packages by Applications. This way, when in need to roll back some changes, it's much easier to keep track and find appropriate Package and corresponding Undo Package related to the application in need of assistance.

Current Plan	Application A	Application B	Application C
Dossiers and Documents	*Not Started*	Under Development	Package_2 for 2018_09_13
Datasets and Components	Under Development	Package_1 for 2018_09_13	Release completed and tested in UAT/QA

Good

Migration Planning

Mega Package for 2018_09_13		
Application D	fixes applied	move all ASAP
Application C	deadline today	move all ASAP
Application B	past deadline	don't move
Application A	needed next week	don't move

Bad

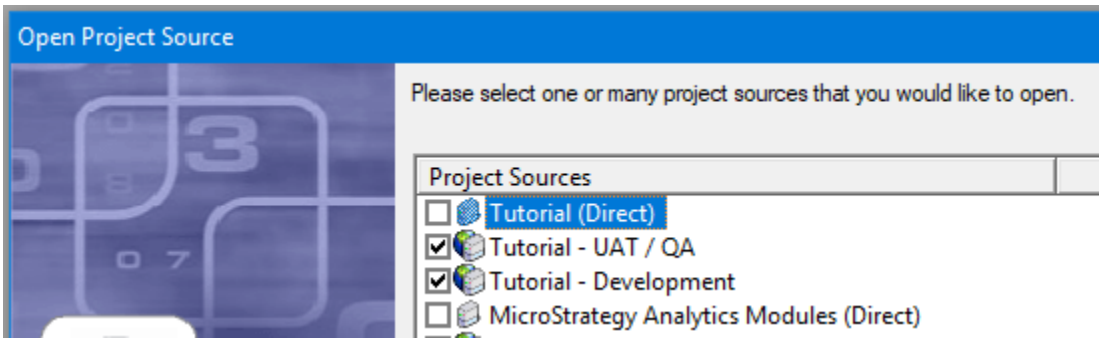
Depending on choosing to build the package by hand, using search objects, folders or using a drag and drop functionality – it may be necessary to further sub-group the objects into packages as used method will determine ability to include more objects in the same package.

This is not a great concern, but then it's important to keep track of order of packages (and their corresponding backups) being created and applied. Reason for that is:

- Packages may share the components – once a component has been synchronized, the second release will re-apply the same version of the object and Undo Package of the second release will not reverse the changes made to that component to its original state.
- Packages may split the components – then order of applying packages matters, due to a dependency that exist in one package and is required for the next package to be deployed.

CONNECTION

Connection to each Environment in Object Manager is done through Project Sources as they've been defined in Developer – it's possible to connect to multiple at the same time:

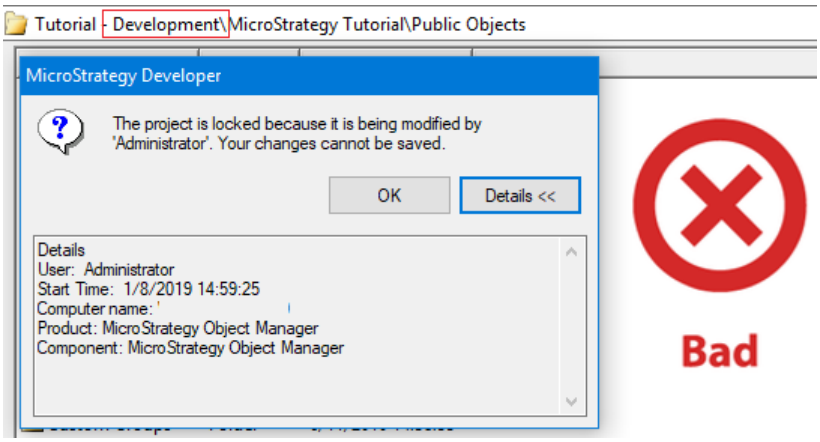


By design, the package will be created from one environment at a time and can also be applied to a single environment at a time.

Avoid Blocking

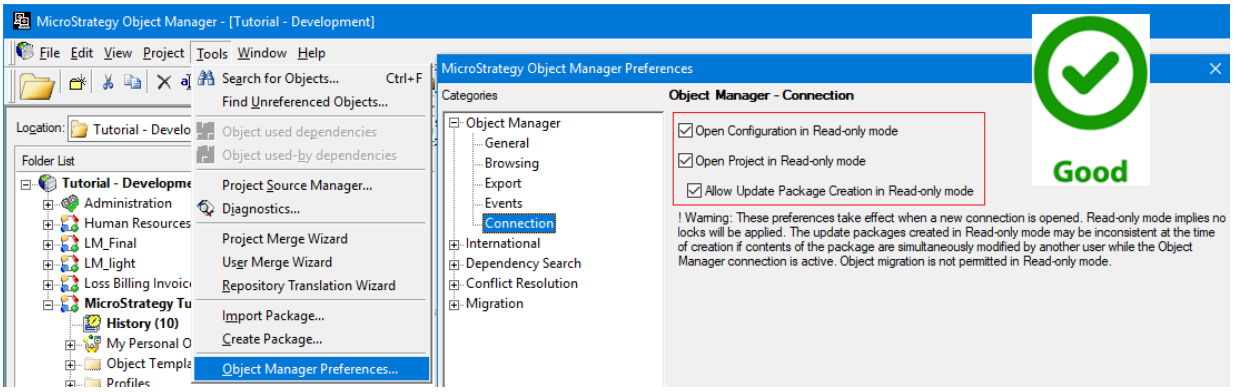
By default, connecting to the environment and opening a MicroStrategy Project with Object Manager will lock the environment and that project which will disable users connected to this project in the same environment from creating or modifying objects.

It's a best practice to avoid locking the Development environment at all times, as there's usually no need to import packages to Development – Platform Administrator would usually connect to Development only to create a Package which does not require putting a lock on the environment or the Project from where the objects will be sourced.



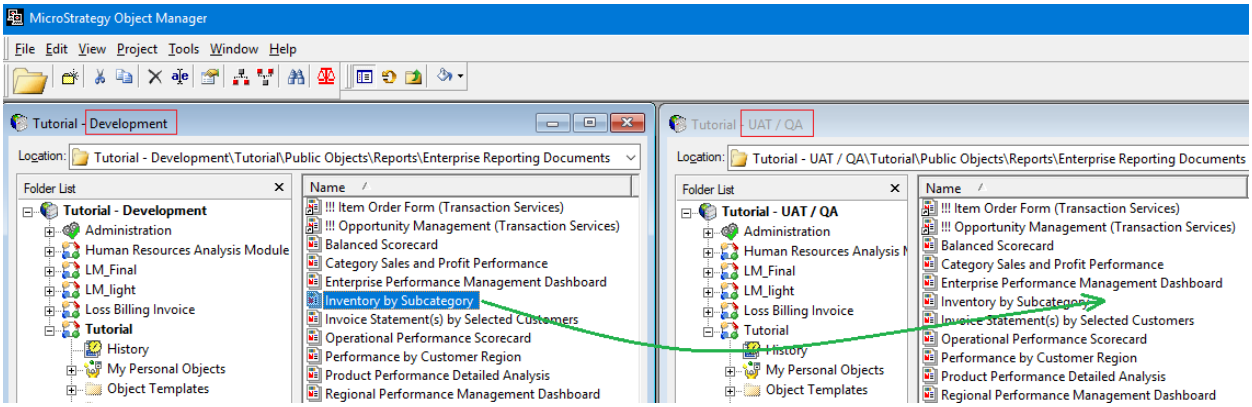
Platform Administrator can take his time when importing packages in UAT/QA as there shouldn't be any development happening there in the first place. However, when connecting to the Development environment to create a package, the Connection settings need to allow

Developers and Architects to continue work on other Applications while the Platform Administrator is drafting his packages for the next Object Migration:

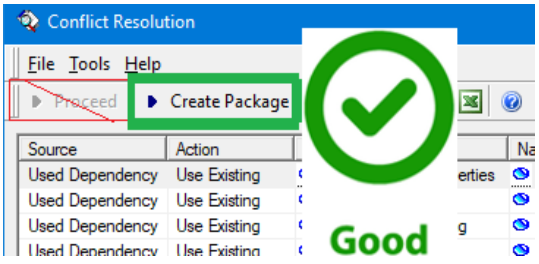


Drag and Drop

It is safe to open two environments at the same time with connection preferences for both Project and Configuration set to “Read-only mode”. This allows to drag and drop high level objects to create and save package that can be later imported into the Target environment.

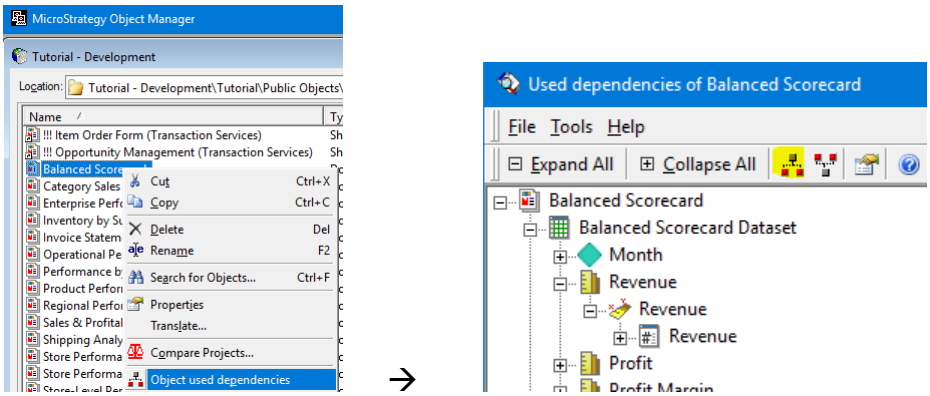


→



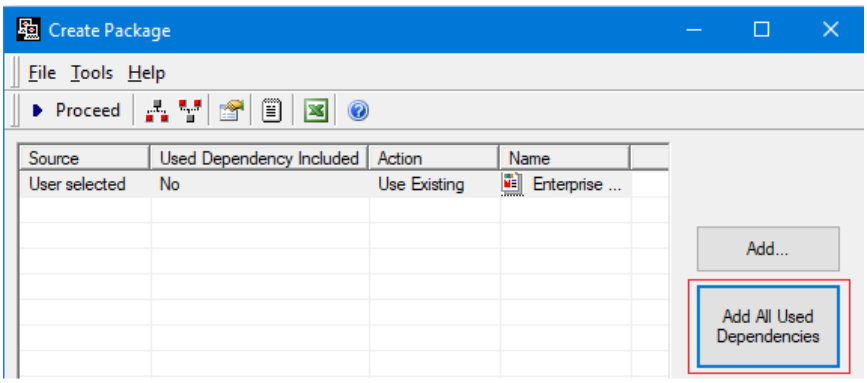
OBJECT DEPENDENCIES

To keep application synchronized, both documents/dossiers and all components of those documents/dossiers need to be synchronized between the environments. Object Manager tool makes it easier for the Platform Administrator to take advantage of dependencies on objects GUID to review all necessary components and dependents of objects:

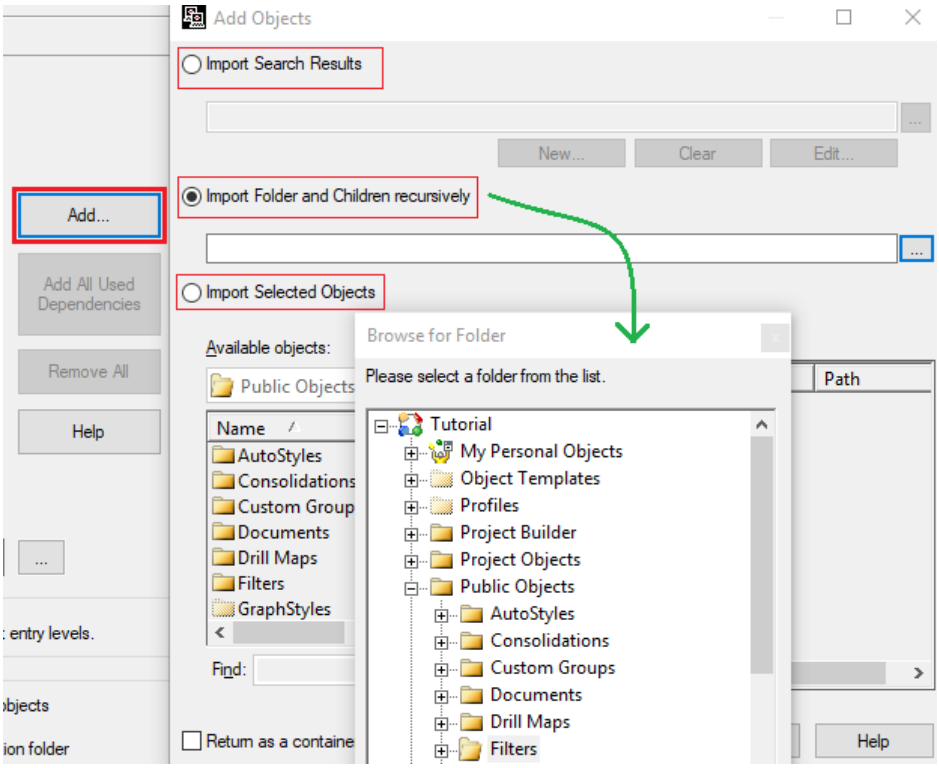


Including all

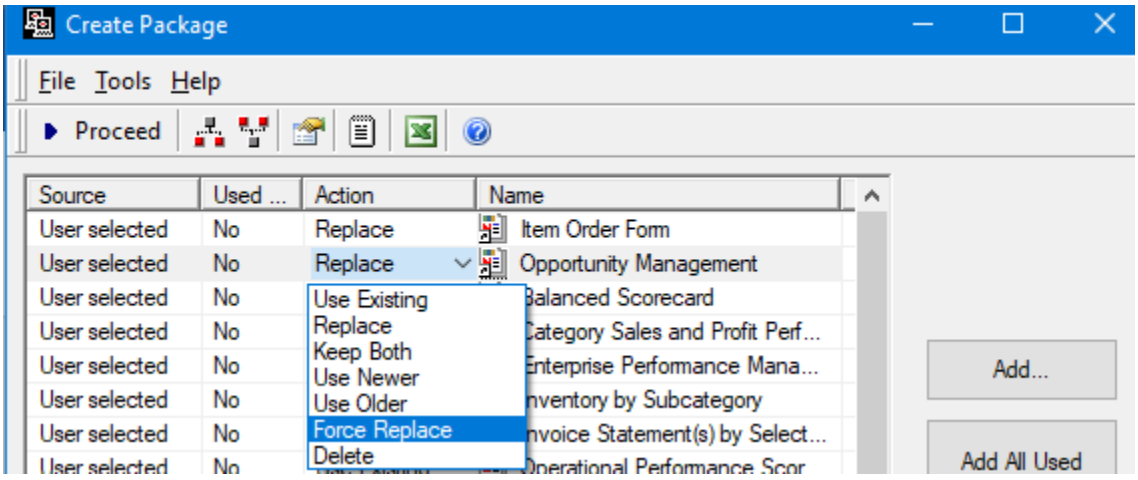
When creating package manually, focusing on Source environment only, Platform Administrator can take advantage of object relationships to include the “All Used Dependencies” of objects placed in the package:



Objects can also be moved by hand from “object used dependencies” or “object used-by dependencies” search window. When clicking on “Add...” button, there will be additional options to insert objects based on the Search object (used extensively with Object Prompts) and based on the Folder – to include all objects inside the folder and all of its sub-folders:

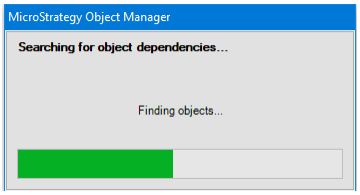


When creating the package manually, it is necessary to set conflict resolution actions basing only on the Source environment. Object included in the package may exist in the Target environment and it could already be in the same version as in the source.

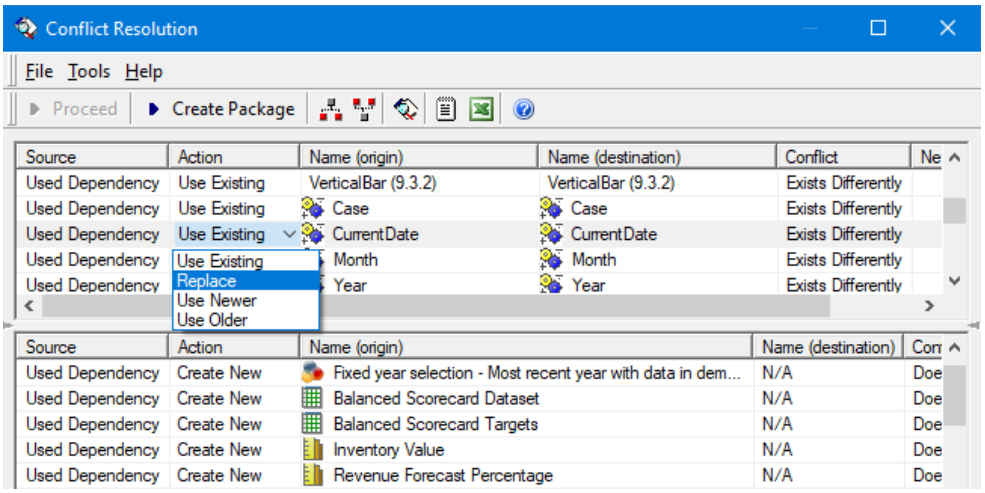


Reviewing a list

When using a drag and drop functionality, the conflict resolution list will be created automatically for all objects selected for the move, as well as all of their components:



All of those objects will be then divided into “New” (bottom part) and “Existing” (top part) based on the Target environment.



For objects already existing in the Target environment it’s possible to choose if desired action is to Replace them or keep using existing version – this can be set for multiple objects at once.

Existing objects will specify if they “Exist Differently” (Version ID is different) or “Exist Identically” (same Version ID) or “Exist Identically Except for Path” – meaning that object was moved to another folder and package can synchronize that, if chosen action is “Replace”.

New objects will always be included and moved without necessity to set any additional options.

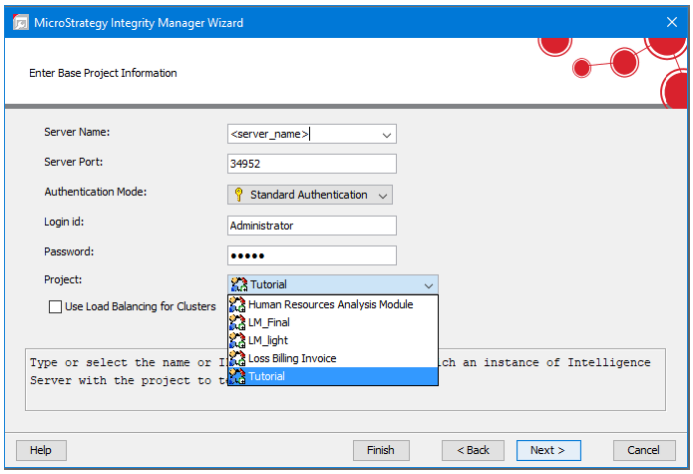
Additional objects cannot be added to the conflict resolution list – that is only functionality of the manual package. Additional objects would need to be included in the initial drag and drop choice or can simply be moved with the next package.

5. INTEGRITY MANAGER

Integrity Manager is another MicroStrategy administrative tool that allows Platform Administrator to run and compare execution results of application objects between different environments.

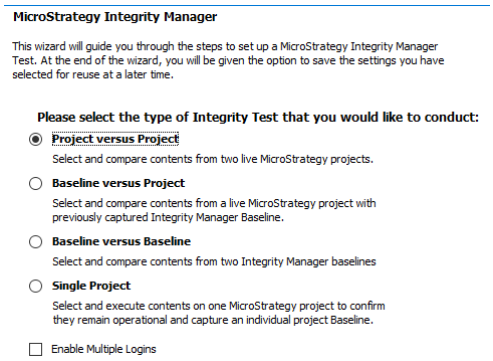
TEST CREATION

To connect to a specific environment, the name of Intelligence Server is required. Reason for not using a Project Sources defined in Developer is due to Integrity Manager capability to connect to different versions of MicroStrategy I-Server for the purpose of testing Upgrades:



Baselining

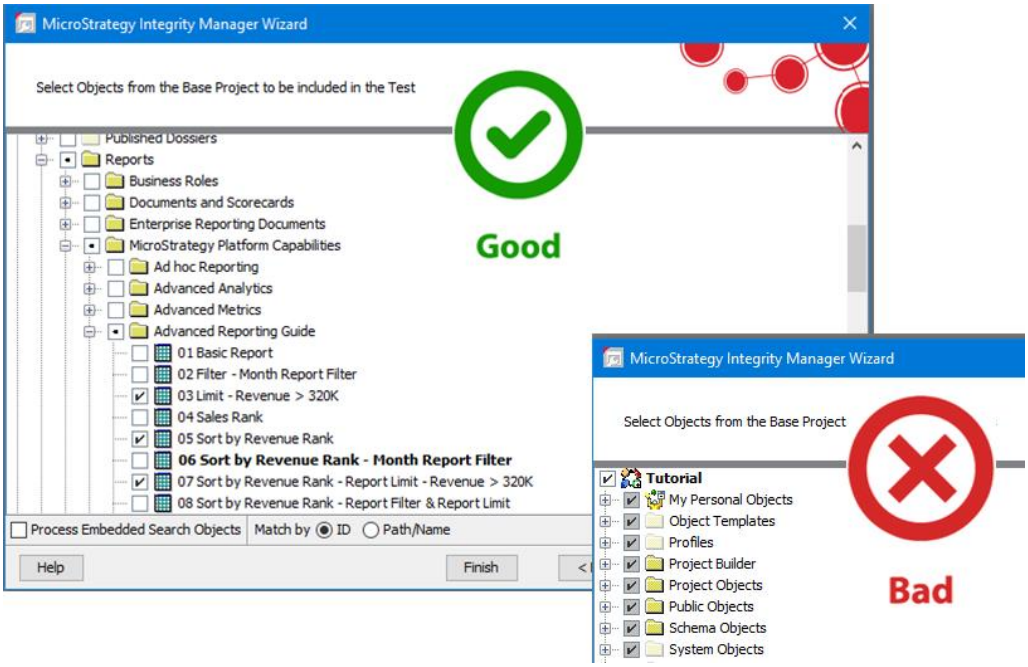
A saved set of results coming from a single environment is referred to as a “Baseline”, therefore there are options to compare Project with a Baseline, two Baselines between each other or just run a Single Project to generate a Baseline – but for the purpose of testing a freshly deployed release it’s usually best to choose Project against a Project:



In case the objects are planned to be further modified right after the Object Manager Package has been created, it's worth considering creating a Baseline in Development environment while building the package. Then Platform Administrator can compare that baseline against live Project in UAT/QA environment once release has been deployed while Developers can work freely on next enhancements.

Selecting Objects

When performing Release Testing of the application, it's necessary to check datasets – Cubes or Grid Reports, that the Application is using as its components. In Integrity Manager this selection is done through check boxes:



It’s possible to select the entire folder which will check all objects inside that folder.

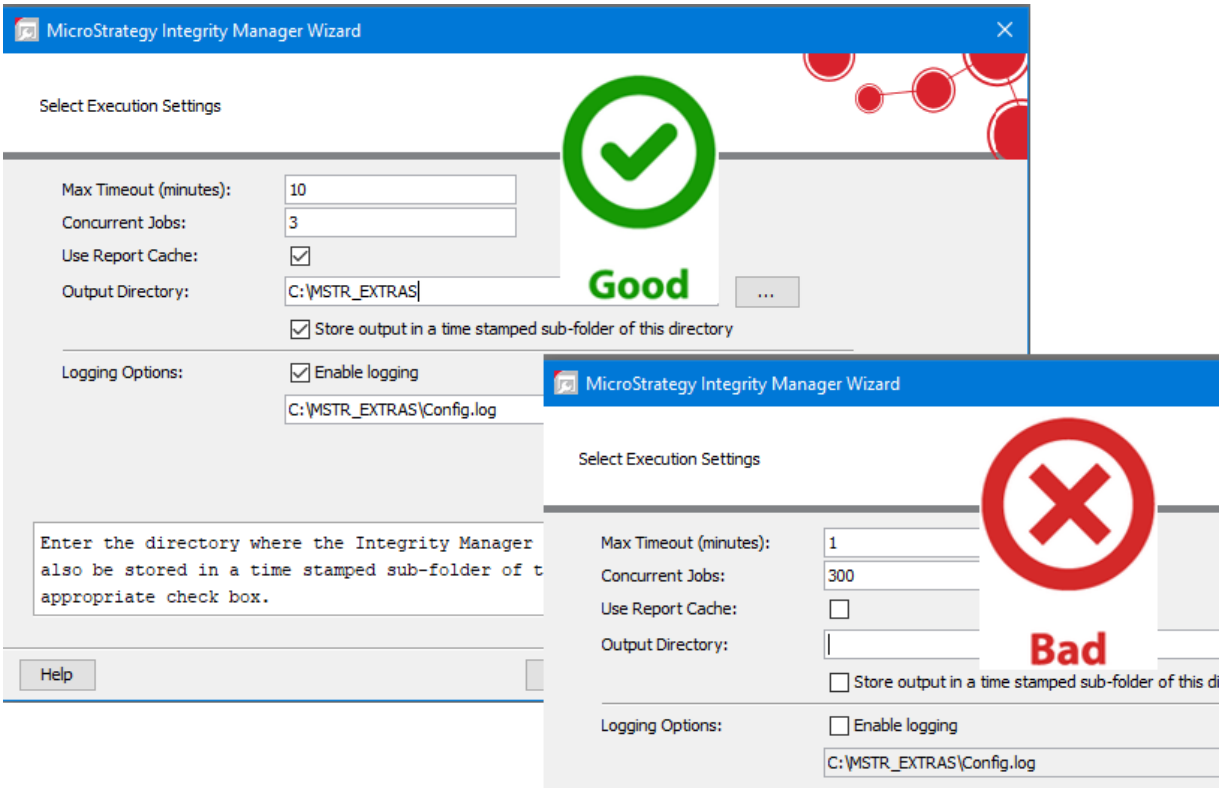
There’s an option to match objects by GUID which the preferred way is. In case there’s a need to compare two datasets with different GUIDs, it only needs to be saved with the same number and under the same path – then, using option to “Match by Path/Name” can be used for the successful configuration of such test.

TEST EXECUTION

Executing the release test should happen possibly as soon as all release packages have been deployed into the UAT/QA environment and it should review objects related to the release.

Settings

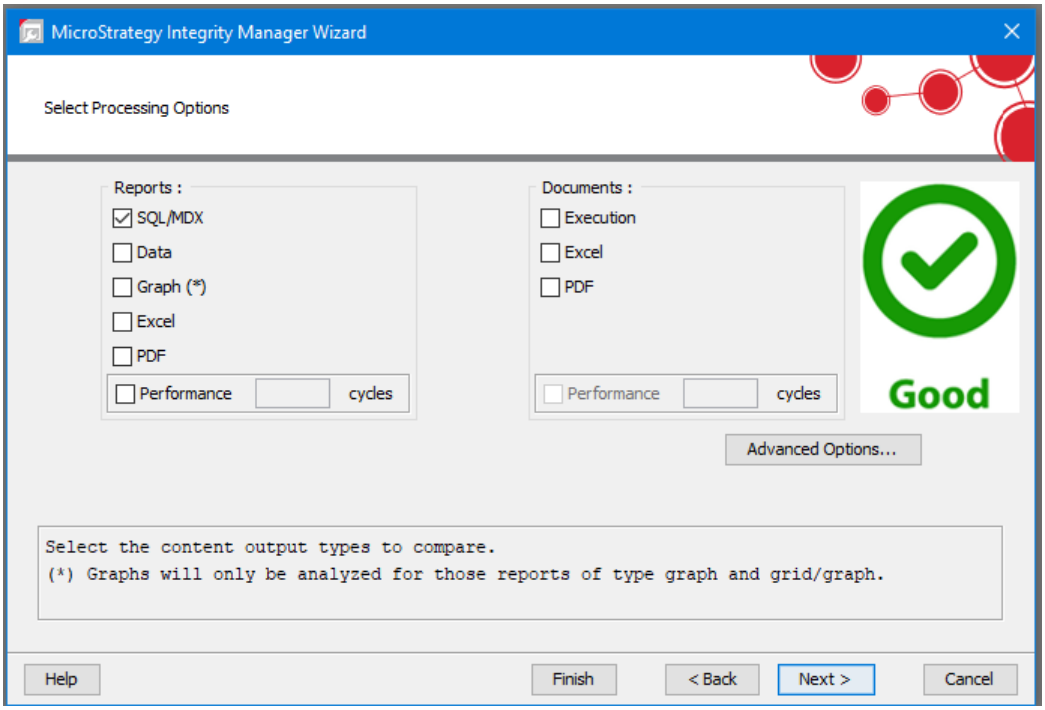
In order to store test results (Baseline) it’s necessary to provide the output directory on the machine where Integrity Manager is being accessed:

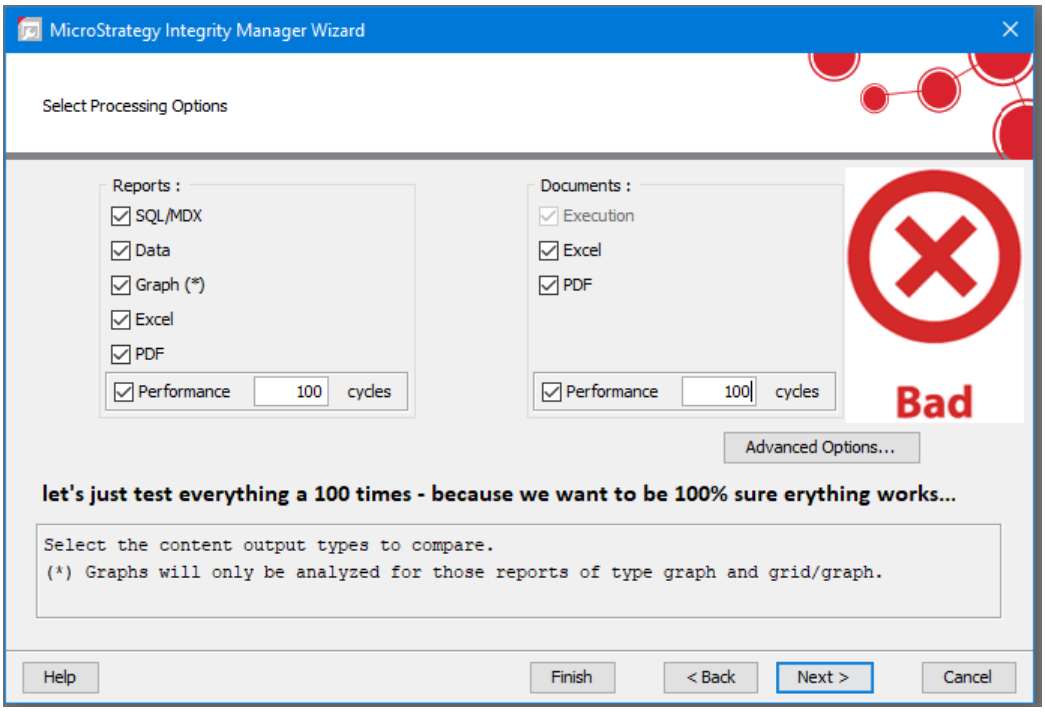


For release testing the Concurrency and Timeout can be set to project defaults or even set lighter. Those settings are for load testing purposes. For release testing, setting should ensure that any failure that comes up will be due to a mismatch between the objects and not due to jobs exceeding the execution timeout limit at a very high job concurrency.

Options

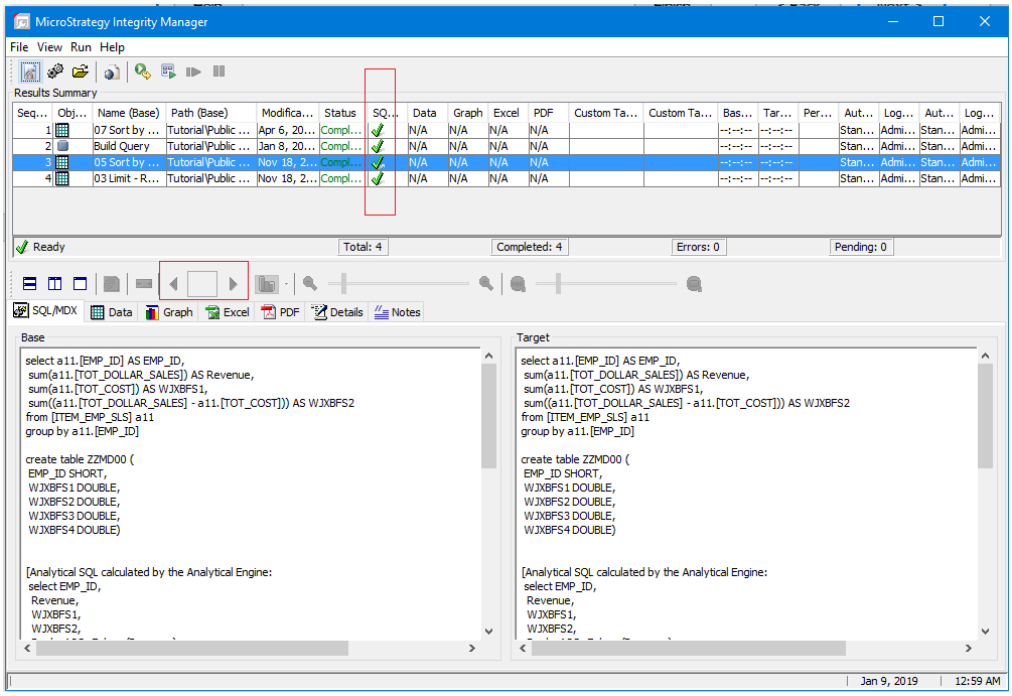
There's usually a different Data Warehouse used by Development and UAT/QA environment. Because of this limitation, comparison of report data will not give any meaningful results. Same goes for the Documents – they will not look the same with different underlying data.





Results

The purpose of release testing is to ensure that objects in UAT/QA environment has been synchronized properly with objects recently created or modified in Development environment. As mentioned in Chapter 2, well synchronized objects should result in the same SQL code when executing the dataset component of the application in both Source and Target environments.



Once the Integrity Manager is finished generating results, Platform Administrator can review the column with execution status for each tested object. Green checkmark means there is a match.

In case of any red cross marks, it's necessary to review differences in results using small left and right arrows. Some differences could be unrelated to the release (example: different default prompt answers between environments) but some could be potential issues coming from a missing piece in the release. Catching small mistakes early in the testing before giving them a chance of becoming serious problems is definitely a good practice.

6. CHANGE CONTROL

Your Intelligent Enterprise must track all planned changes through a series of checklists, analyses and approvals to ensure your business is fully prepared to implement any operational change.

Throughout the object migration process, establish and adhere to a consistent change management system. This may include creating tickets for individual objects or for groups of objects.

Formally review the objects for quality and for contribution within the project.

Use the change control process to authorize object migrations and track their promotion. In the MicroStrategy 'Change Journal', add change management artifacts to track changes during review.

Naming convention is also very important. Name the Object Manager package files the name/number of the change control ticket and keep an archive of those files. This will help with tracking and migrating the correct objects authorized by change control.

7. USEFUL LINKS

OBJECT MANAGER

https://www2.microstrategy.com/producthelp/current/SystemAdmin/WebHelp/Lang_1033/Content/Copying_objects_between_projects__Object_Manager.htm

INTEGRITY MANAGER

https://www2.microstrategy.com/producthelp/current/SystemAdmin/WebHelp/Lang_1033/Content/Integrity%20Manager.htm