

Application Version 1.1

FAQ's

- Is integration real-time?
 - It is not. The tool follows a 'train schedule' like the design.
 - Assuming the repeat interval of the jobs is 10 minutes. The first train would update the proposed path of the components in the component tracker. The duration of this activity would be around 0-10 minutes.
 - The next one would commit the update sets. The duration of this activity would be around 0-10 minutes.
 - Adding the one before, the first update set would be committed after 20 minutes and the second update set would be committed after 30 minutes and so on.
 - If you missed the first train, you would have to wait for the next one and because of this, the repeat interval of the job becomes crucial.
 - Below diagram can provide more explanation

	All updates loaded in QA					
10.00 AM		Update set 1 committed				
10.10 AM			Update set 2 committed			
10.20 AM				Update set 3 committed		
10.30 AM					Update set 4 committed	
10.40 AM						

Assumptions:

- Time duration of the job is 10 minutes

No errors/warning found

- Connectivity is established and working between prod and QA and QA and dev.

- All update sets are loaded in the first go.

-Update sets get committed flawlessly within in 10 minute duration

- Update (as of 2-Jan-2020): As there was slowness observed, there have been some changes made to the above logic. The design stays the same as above but at each execution, the function calls are called 5 times. This is analogous to 5 trains appearing at the platform every 10 minutes instead of 1. This way there will be at least 2 to 3 update sets committed for every job execution.
- Does this tool support batching of update sets?
 - As of 24-Dec-2019, it does not.

- What is the Timezone used to evaluate the Change Request window?
 - The time zone of the user profile which is used to Run the 'ServiceGuru - Auto Update Set and Code Promotion' job.

The screenshot shows the 'Scheduled Script Execution' configuration page for 'ServiceGuru - Auto Update Set and Code Promotion (Turn ON in Orchestrator Instance ONLY)'. The 'Run as' field is highlighted with a black box, showing the user 'SN Guru Automation a' with an information icon. Below this, the 'User' profile is displayed with various fields. The 'Time zone' field is highlighted with a black box, showing 'Canada/Eastern'.

Configuration Details:

- Name:** ServiceGuru - Auto Update Set and Code Promotion (Turn ON in Orchestrator Instance ONLY)
- Active:** ☒
- Application:** Auto Deployment - ServiceGu
- Run:** Periodically
- Repeat Interval:** Days: 0, Hours: 00, 00, 10
- Run as:** SN Guru Automation a
- User Profile:**
 - User ID:** my_test_integration
 - First name:** SN Guru
 - Last name:** Automation account
 - Title:**
 - Department:**
 - Password:**
 - Email:**
 - Language:**
 - Notification:** Enable
 - Calendar:** Outlook
 - Integration:**
 - Time zone:** Canada/Eastern
 - Date format:**

- **KNOWN ISSUE:** I am looking to deploy my components from TEST to Production and for that, I opened the catalog, selected the release and I changed the destination environment to Production and tagged a Production Change Request but I could not submit the request as it gives the below message.

No Update Set or Script change detected. Please change an entry and submit again.

OK

- This issue can be resolved by adding more code and at this point it was considered not worth the effort.
- The easiest workaround is to slightly change the order of any components and submit the request. Example: Change the order from 20 to 21.

- What is the direction of code movement?



Environment Definition

* Dev Instance

-- None --



QA Instance

-- None --



UAT Instance

-- None --



Pre-Prod

-- None --



Intermediate Instance 2

-- None --



Intermediate Instance 5

-- None --



Intermediate Instance 3

-- None --



Intermediate Instance 4

-- None --



Intermediate Instance 6

-- None --



Intermediate Instance 7

-- None --



Intermediate Instance 8

-- None --



Intermediate Instance 9

-- None --



Intermediate Instance 10

-- None --



Intermediate Instance 11

-- None --



Intermediate Instance 12

-- None --



Intermediate Instance 13

-- None --



Intermediate Instance 14

-- None --



Intermediate Instance 15

-- None --



Intermediate Instance 16

-- None --



Intermediate Instance 17

-- None --



Intermediate Instance 18

-- None --




Intermediate Instance 19

-- None --




- In the following sequence: Whichever instance (example acmedev1) is selected as Dev instance will act as the source for a given Release and all update sets for that release are pulled from acmedev1. Deployments can never be done to Dev instance as it's the source.
- The next environment will be the one selected in the QA Instance (Example: acmeqa1). All update sets pulled from acmedev1 are committed to acmeqa1.
- From acmeqa1, the update sets/scripts are committed to instance selected in UAT (acmeuat1). From UAT it flows to Pre-Prod (Intermediate Instance 1). From Pre-Prod (Intermediate Instance 1) the components next move on to Intermediate Instance 2. From Intermediate Instance 2 the components next move on to Intermediate Instance 3 and so on.
- Finally, the last environment for all components is always production or whichever environment is captured in the system property 'x_7756_update_set.service_guru_production_instance'.
 - If you intend to just use this tool within just non-production instances, example: DEV, QA and UAT, Enter the UAT instance name in the x_7756_update_set.service_guru_production_instance property instead of the production instance.

Enter the production instance name. 

acmeuat1

- Also, choose acmeuat1 as the Orchestrator instance.

Enter the instance which will orchestrate all the commits. Ideally, this would be the production instance but it can also be a non-prod instance. This property will be useful in a non-production environment to ensure that the scheduled job which orchestrates all the commits not accidentally run in an environment that is not designated as the orchestrating environment. (Ensure that all non-production environments have the same value for this specific property where this tool is installed) 

acmeuat1

- Similarly, UAT (acmeuat1) credentials need to be updated in the below properties.

Production Only - This property sets the username for the User Profile used for Integration for the Production instance defined in the property `x_7756_update_set.service_guru_production_instance`. [?](#)

Production Only - This property sets the Password for the User Profile used for Integration for the Production instance defined in the property `x_7756_update_set.service_guru_production_instance`. [?](#)

- How does the Order field work?
 - The order decides the execution of components relative to the release. It is rank wise and a component with a smaller order will be executed first.

Order	Execution sequence	Update set
10	First	My-release-update-set-v1
20	Second	My-release-update-set-v2
30	Third	My-release-update-set-v3
40	Fourth	My-release-update-set-v4

- If due to any reason, if a component in the middle of the sequence is stuck. Example:
To resolve errors/warnings, the components with higher-order will not be executed.

Order	Execution sequence	Execution status	Update set
10	First	Committed	My-release-update-set-v1
20	Second	Stuck (Waiting for errors/warning to be resolved.)	My-release-update-set-v2

30	Third	Stuck (Waiting for My-release-update-set-v2 to be committed)	My-release-update-set-v3
30	Fourth	Stuck (Waiting for My-release-update-set-v2 and My-release-update-set-v3 to be committed)	My-release-update-set-v4

- Components are strictly deployed as per the order stored in the Tracking table.
 - The exception to this rule: For components of the same release, Scripts are **ALWAYS** committed after all update sets are deployed irrespective of the order of a script compared to the order of the update set.

Example: If the below components are to be committed...

Order	Component type	Update set	Script
10	Update set	My-release-update-set-v1	
20	Script		gs.log('test log');
30	Update set	My-release-update-set-v2	

Execution sequence will be as below:

Order	Execution sequence	Component type	Update set	Script
10	First	Update set	My-release-update-set-v1	
20	Third	Script		gs.log('test log');
30	Second	Update set	My-release-update-set-v2	

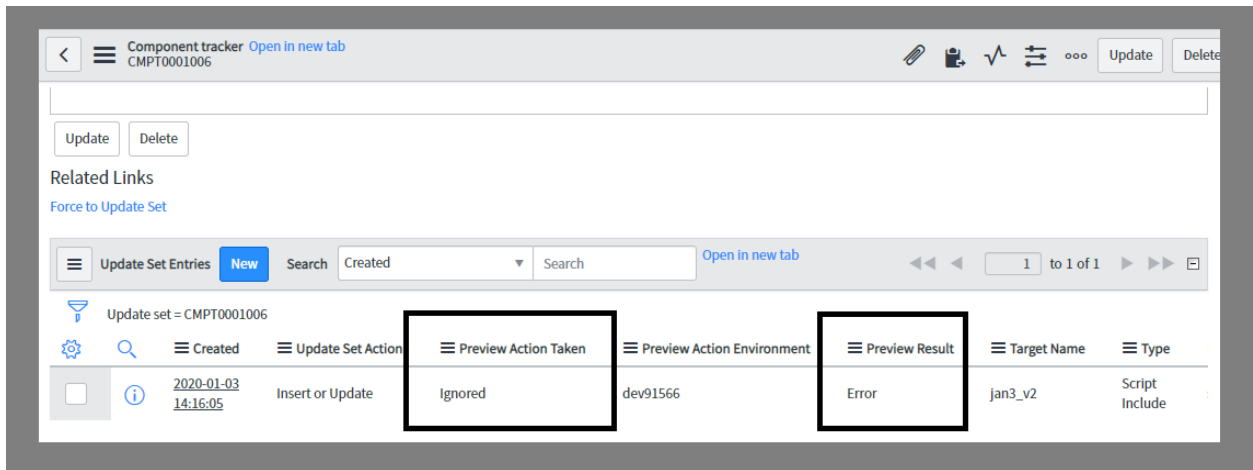
- Another reason for a component to be stuck is if the component with a lower order of a different release is stuck in a given environment. Refer below table for more explanation

Order	Instance of commit	Release	Execution sequence	Execution status	Update set
10	Production	ABC	First (in Prod)	Committed	My-release-abc-update-set-v1
20	Production	ABC	Second (in Prod)	Stuck (Waiting for errors/warning to be resolved.)	My-release-ABC-update-set-v2
30	Production	XYZ	Third (in Prod)	Stuck (Waiting for My-release-abc-update-set-v2 to be committed)	My-release-xyz-update-set-v1
40	Production	XYZ	Fourth (in Prod)	Stuck (Waiting for My-release-abc-update-set-v2 and My-release-xyz-update-set-v1 to be committed)	My-release-xyz-update-set-v2
50	TEST	ABC	First (in TEST)	Committed (does not get impacted by execution status of components of a lower order of 20 or 30 as this update set is the first to be committed in TEST)	My-release-ABC-update-set-v3
60	TEST	ABC	Second (in TEST)	Stuck (Waiting for errors/warning to be resolved.)	My-release-ABC-update-set-v4

70	TEST	xyz	Third (in TEST)	Stuck (Waiting for My-release-abc-update-set-v4 to be committed)	My-release-xyz-update-set-v3
80	TEST	xyz	Fourth (in TEST)	Stuck (Waiting for My-release-abc-update-set-v4 and My-release-xyz-update-set-v3 to be committed)	My-release-xyz-update-set-v4

- Why the order updates that I do via a service request, do not correctly reflect in the tracking table?
 - As discussed above, Orders are used to not only execute components within the same Release but for a given environment if 2 different releases are to be committed together, the components with lower order will be executed first.
 - From the Service Catalog, you may not get a complete picture of the order of components for a given release relative to other ones and so it is better to update the Order via the tracking table.
- We cloned our Development environments from production. Can we use this tool to commit the components in the Development environment?
 - Yes, but it is not recommended because this tool is not designed for this use case but you can try a workaround.
 - Submit a Service Request or directly make the following changes
 - Make the Development environment as the QA and the QA as the Dev. This way, the components will be pulled from the QA environment and committed to the Development environment.
 - Important: Ensure that there is an update source in the Development instance with the source as the QA instance.

- Once the components are committed in the Development environment, switch back the two instances.
- Can this tool support more than the existing 19 intermediary instance?
 - The design, unfortunately, is not flexible at the moment but it is easy to add another 20th instance. You would need some scripting knowledge for the same. Search all the components with the keyword 'intermediate_instance_' and you would be able to figure out where to make the changes to. To perform a global script search, open Studio in ServiceNow and search in all application scopes.
- I have questions about the error acceptance mode property
 - There are 4 different modes to deal with issues that are seen after previewing an update set: Manual, Smart accept, Accept all, Skip all.
 - More details:
 - Manual (default): In this mode, the errors and warnings are to be resolved by the developers. Once the errors and warnings are resolved, the update set is committed.
 - Smart Accept:
 - If the code movement is from DEV to QA, QA to UAT and UAT to Production, and if the tool sees any errors or warning while previewing in the QA instance, the tool waits for the developers to manually resolve these issues.
 - Now for the UAT deployment, if the same errors and warnings are seen again, the tool will repeat the actions performed by the developers in QA. This way, the developers do not need to manually intervene to repeat the same activity. This data is stored in the Related table called Update Set Entries (x_7756_update_set_update_set_entries) and is a related list to the Component Tracking table.



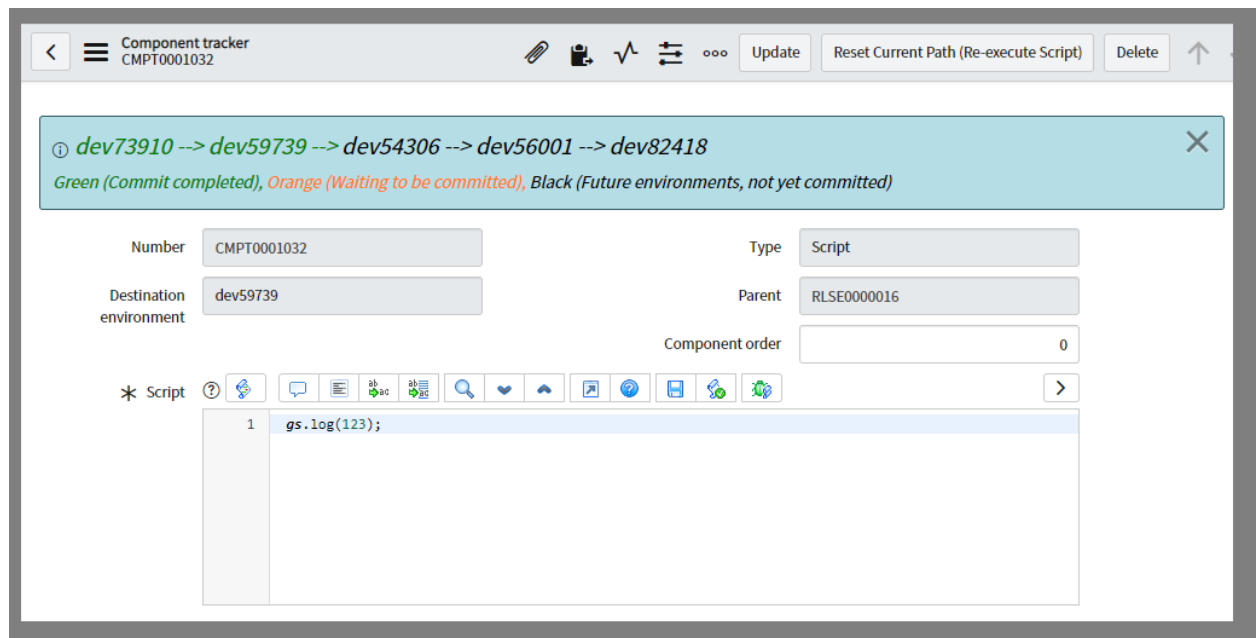
- Now for Production deployment, if the same error/warning is seen which were observed in the UAT instance, the actions performed in the UAT instance will be repeated.
- If a new error or warning is found in either UAT or Production deployment which the tool did not record before, **the tool will wait for the error/warning to be resolved manually** and then commit the update set. The tool will capture the action taken against this new error or warning so that it can resolve the issues in the next environment.
- Because of the table design, at a given point in time, the actions of the previous instance are stored in the tool and the actions for the ones before the previous are lost. Example: If the deployment is about to be done for a Production instance, the actions taken in the UAT are stored and the ones of QA are overwritten and are no longer directly available. Although as Audit is enabled, you can check the history to see what knowledge was stored for the QA instance.
 - Accept All: In this mode, the errors and warnings are auto-accepted by the tool.
 - Skip all: In this mode, the errors and warnings are auto-skipped by the tool.
- Whichever of the above 4 modes is selected is global for all the commits that happen. This means the same model will be applicable for components of all the

Releases and all the different non - production and production environments. So you cannot pick and choose 'Smart accept' mode for UAT deployments and 'Accept all' mode for Production deployments. Similarly, you cannot select 'Smart accept' mode for deployments of components for Release ABC and 'Accept all' mode for deployments of components for Release XYZ.

- Can the tool send emails to notify about the errors?
 - ~~No, because of the limitation of the design, the way it works is that it would frequently generate lots of errors repeatedly and it would clog your inbox with duplicate messages, its best to leave these messages in logs. If you wish to, you can modify the 'this.info' functions in the script includes to also send an email along with the logs (not recommended).~~

As on 24 Dec 2019, we have introduced the digest email functionality which aggregates the logs and sends them as emails to the Operations team.

- How are multiple scripts committed? Does the tool wait for one script to be committed first and then commit the second?
 - There is no wait time between the 2 scripts and it is highly recommended to combine all the scripts into one single script so that the execution is predictable or the tool will do this for you (in an unpredictable fashion based on the order).
 - If you cannot combine all scripts in one, **encapsulate the script in a function so that the results are predictable.**
- How do I commit a specific script again in all the environments?
 - Delete the scripting component from the tracking table and create a new entry
 - Alternatively, open the specific tracking record for the script you intend to commit again and click on 'Reset Current Path'. This would cause the script to be committed in the destination environment and all the environments before that.



- Does the tool support individual unique credentials for each environment?
 - Yes. By default, the password of all non-production environments is the same but you can override this by creating custom properties under this custom application scope.
 - For example, your ServiceNow instance name is 'ACME' and you have 2 Dev (acmedev1, acmedev2) and 2 QA (acmeqa1, acmeqa2) environments, and you would like to create separate credentials for qa1 and qa2. You need to create below 4 properties **under the custom application 'x_7756_update_set' scope** :
 - username_of_acmeqa1
 - password_of_acmeqa1
 - username_of_acmeqa2
 - password_of_acmeqa2
 - Note, the above properties need to be created in the orchestrator instance.
- How does the tool deal with update sets which were loaded quite a while back? Does it run preview again?

- The tool checks if the update set which is about to be committed was loaded BEFORE the current hour. If it was, the tool deletes it and **loads and previews the same update set again**. The tool will wait for this load to complete before committing this update set. Example: If the job executes at 5.35 PM on 1/1/2020 then the update sets loaded/previewed AND which are tracked by this tool, which was created before 5 PM will be deleted. Refer below table for more explanation. Here 'Some-Other_project-v1' update set is not tracked by the tool whereas other update sets are.



Retrieved update set name in a TEST	Time of update set retrieval in TEST	Is the tool tracking this update set	Operation
Some-Other_project-v1	1/1/2020 4.45 PM	No	No deletion
My-tracked-project-v1	1/1/2020 4.55 PM	Yes	Delete
My-tracked-project-v2	1/1/2020 5.05 PM	Yes	No deletion
My-tracked-project-v3	1/1/2020 5.15 PM	Yes	No deletion
My-tracked-project-v4	1/1/2020 5.25 PM	Yes	No deletion

- Does the tool deal with loaded update sets which I manually uploaded in any environment?
 - The tool checks if the update sets were loaded via a specific update source and if the update set was instead manually loaded, the tool will not commit it. It will instead wait for the update set to be loaded from the expected update source.
 - Also, the tool at the moment only deals with previewed update sets and not with 'loaded' ones.
- How do I onboard a new production instance into this tool?
 - Follow all the prerequisite steps listed in the installation document.
 - Install the update sets for this tool in this new non-production instance.


- Follow all the steps listed in the installation document for a non-orchestrator instance.
 - Define update sources in this new environment if we want to bring in Update sets from another instance. Similarly, if this instance acts as a source, for instance, XYZ, define an update source in XYZ for this new production instance.
 - In the orchestrator instance, Add this new instance in property `x_7756_update_set.non_prod_us_deployment_environments`
 - If this instance needs a unique set of credentials for the integration, refer to the above section on how to add custom credentials. This step needs to be performed in the Orchestrator instance.
- Why does the tool not preview the update sets and instead rely on `'glide.update_set.auto_preview'` to be true?
 - The tool relies on ServiceNow to auto preview and not preview the update sets by itself so that it's faster and a lot less complicated overall with maintenance in mind.
 - Also, if the preview logic in the future is changed by ServiceNow, this design makes it future proof w.r.t ServiceNow platform upgrades.
- As the tool does not support the execution of scripts in specific application scope, how do I do this if I want to?
 - At the time of writing this, the developers are not aware of a flexible way of handling this but there are ways around this.
 - Capture your script in a scoped script include. Ensure that the `'Accessible from'` is set to `'All application scope'`.

<

Script Include
New record



Name

my_scope_app_script_include 


API Name

x_7756_update_set.my_scope_app_sc

Client callable

☐

Application

Some custom App Scope 

Caller Access

-- None --

Accessible from















All application scopes

Active

☒

Description

Script



```
1 var my_scope_app_script_include = Class.create();
2 my_scope_app_script_include.prototype = {
3   initialize: function() {},
4   execute: function() {
5     // write your code here.
6   },
7   type: 'my_scope_app_script_include'
8 };
```

- Capture this above script in an update set and add it as part of the deployment.
- Add a script in the deployment as below.

Add Row

* Order

20.0

Update Set Name

* Script

```
my_function();

function my_function(){

var my_scoped_obj = new x_7756_update_set.my_scope_app_script_include();
my_scoped_obj.execute();

}
```

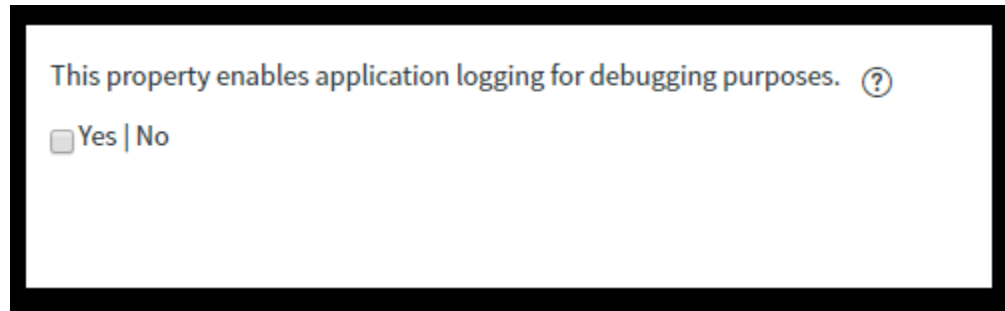
Component Changed

Yes

- Make sure you have a process to clean up these scripts includes post-deployment.

Troubleshooting

- Check the frequency of the job. If the repeat interval is hourly or more, update sets would be committed at a very slow pace (maybe 1 every 1 or 2 hours). Increasing frequency can help. The ideal value of the repeat interval should be every 10 minutes.
- Enable the below property if you want to see some errors related to integration, connectivity, etc. Logs are captured in application logs (syslog_app_scope) and syslog



- Update sets not being committed?
 - Connectivity issue: Check user name and password and verify the connectivity.
 - Check if the ID used to commit update sets has minimum rights to perform this activity (admin rights are preferred).
 - Check if there is a valid update source which is bringing in the update sets
 - If you see the update set in a previewed state and not being committed, chances are that the update set was brought from a different instance (by a different update source) and the tool has a check to see if the update set was brought from the correct source instance. Example: If the update set 'XYZ' was brought in QA instance from dev2 update source and if the tool expects the source instance to be dev1, then this update set will not be committed.
 - For a specific target environment, the job checks if any components are backed up. Check the order field and see the first component is waiting to be committed. Once the issue with that component is fixed, other components should start working. These messages are also recorded in the application logs table of the orchestrator instance (if logging is enabled).

- One of the reasons for partial update sets being committed could be because the change window duration was not long enough compared to the number of update sets. Example: If the change window was for 1 hour and the repeat interval was for every 1 hour, the maximum number of update sets that could be committed is possibly 2-3 because the job just got one chance to execute within the change window. If the release instead had 20 update sets, chances are not all update sets will be committed. To fix this, either increase the repeat interval frequency of the scheduled job or have a longer duration for the change window.