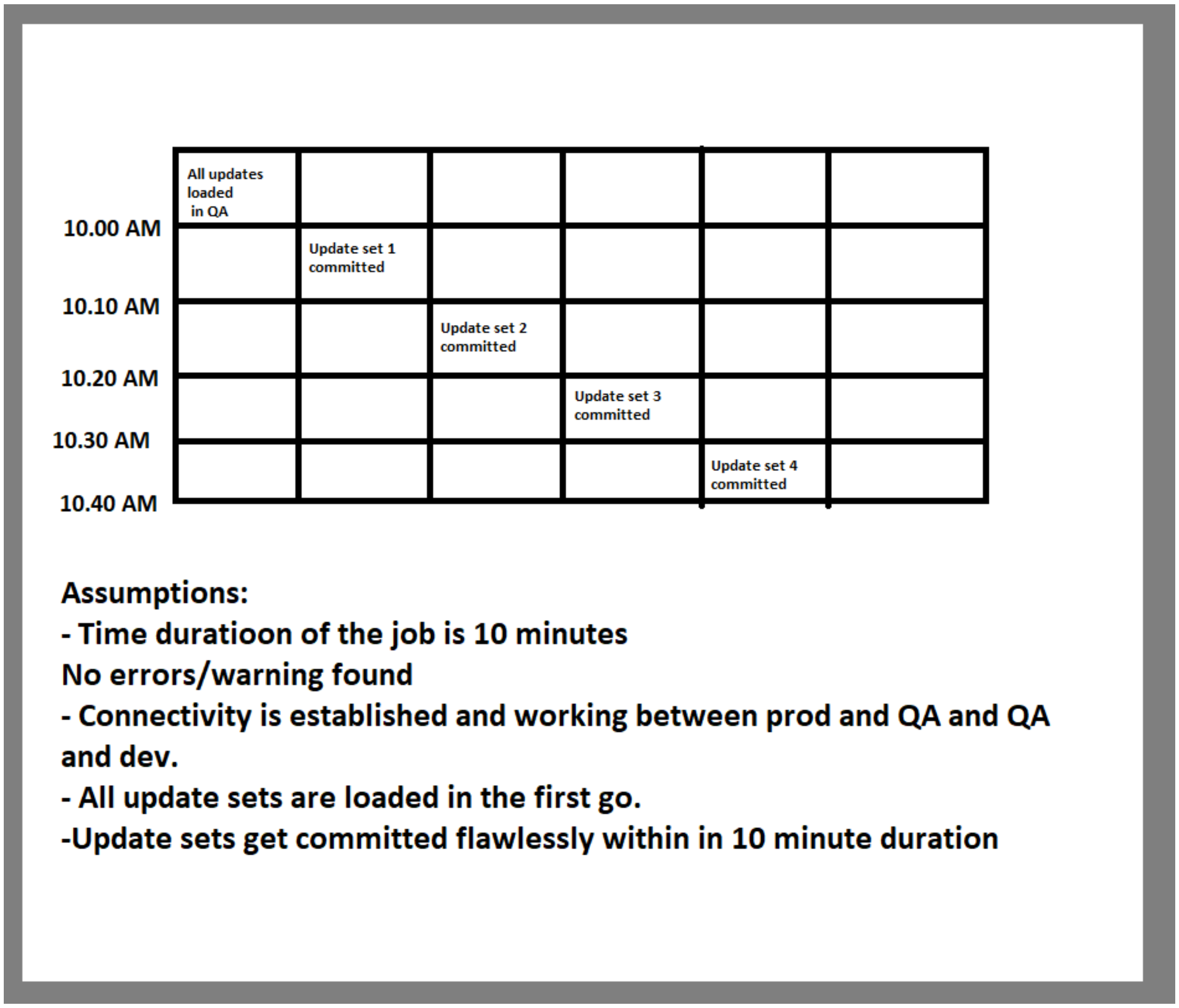
Application Version 1.1

FAQ’s

- Is the tools integration real-time?
    - It is not. The tool follows a ‘train schedule’ like 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



Assumptions:

- Time duratioon 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 hierarchy of the environments?
    - In the following sequence: Dev, QA, UAT, Pre-Prod (Intermediate Instance 1), Intermediate Instance 2, etc.

- We cloned our dev environments from production. Can we use this tool to commit the components in the dev 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 dev 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 dev environment.
    - Important: Ensure that there is an update source in the dev instance with the source as the QA instance.
    - Once the components are committed in dev, switch back the two instances.

- Can this tool support more than the existing 19 intermediator 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
    - The property has 4 different values
        - 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.

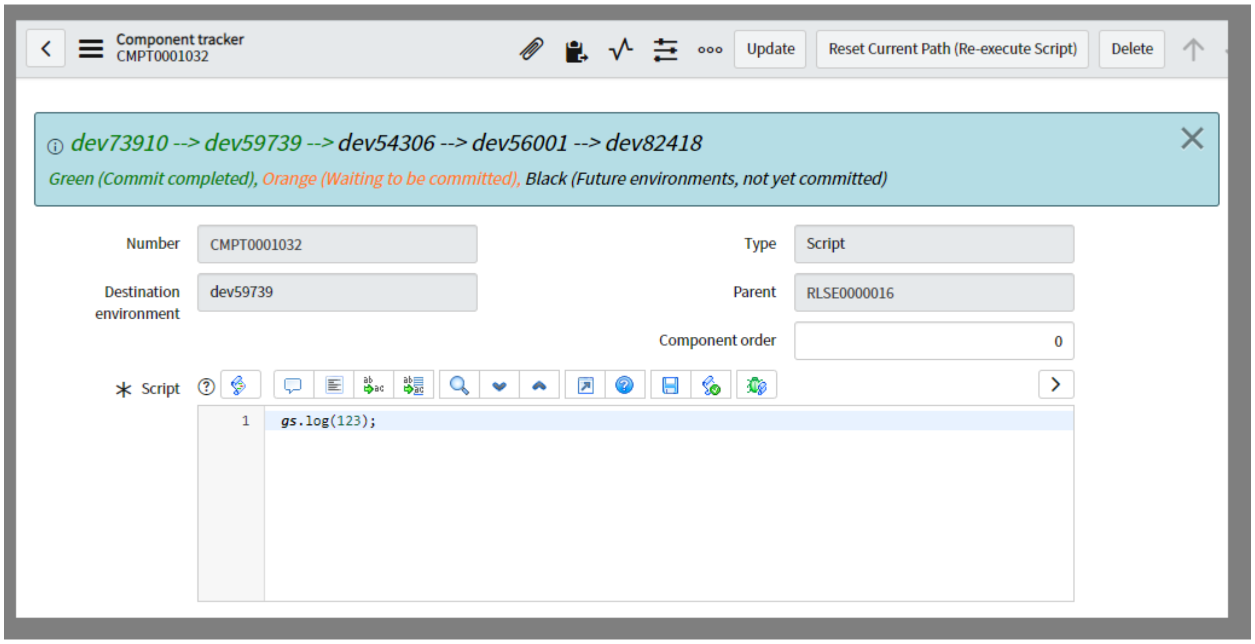
        - Smart Accept: In this mode, the errors and warnings are accepted or skipped based on the decisions taken in the previous instance. If for example, you see the same error in UAT and you had already accepted that in QA, the tool will auto-accept the error for you. Similarly, if the action in QA was to skip the error, the tool will skip it. If a new error or warning is found, the tool will wait for the error/warning to be resolved manually and then commit the update set.
        - 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.

- 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 basically 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.



dev73910 --> dev59739 --> dev54306 --> dev56001 --> dev82418

Green (Commit completed), Orange (Waiting to be committed), Black (Future environments, not yet committed)

Number	CMPT0001032	Type	Script
Destination environment	dev59739	Parent	RLSE0000016
		Component order	0

Script

```
gs.log(123);
```

- 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, acmeqa1) 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

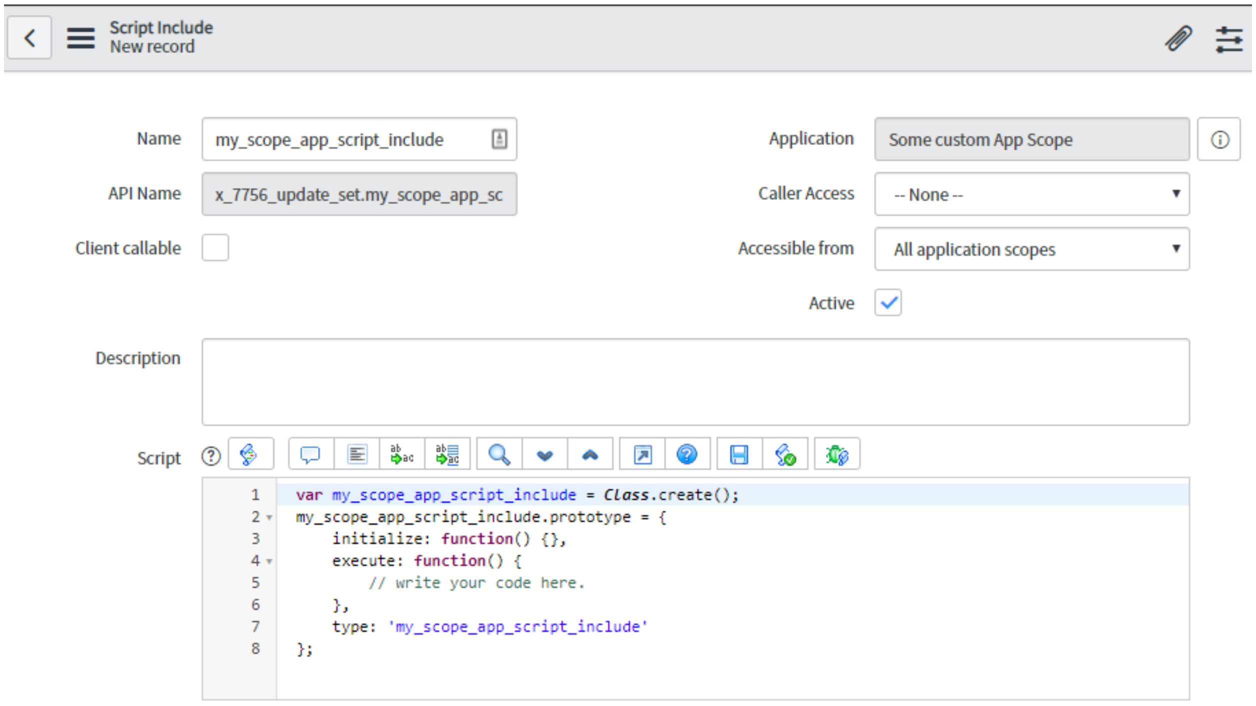
        - password_of_acmeqa2
    - Note, the above properties need to be created in the orchestrator instance.

- 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 in the current hour. If it was, the tool deletes it and loads and previews the same update set again.
- 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.
    - 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 pre requisite 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 the above section on how to add custom credentials. This steps needs to be performed in the Orchestrator instance.

- Why does the tool not preview the update sets and instead relies 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 future is changed by ServiceNow, this design makes it future proof w.r.t upgrades.

- As the tool does not support execution of scripts in a 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'.

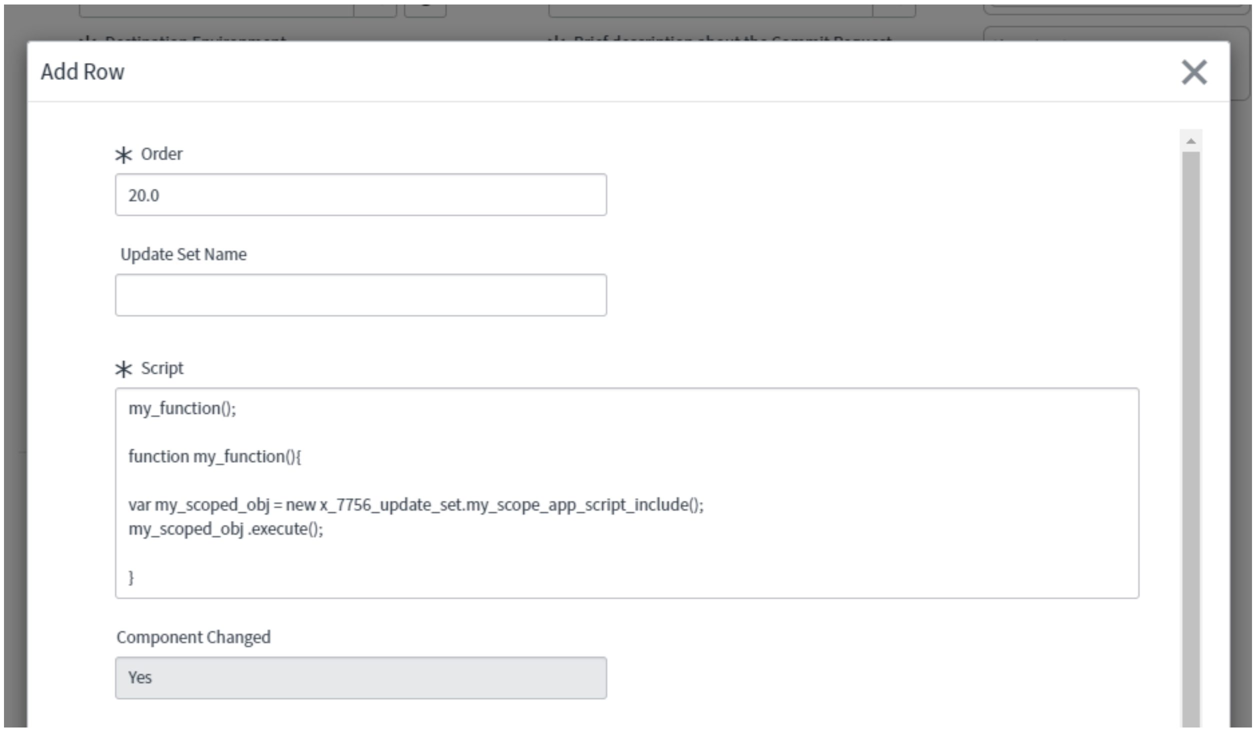


Name	my_scope_app_script_include	Application	Some custom App Scope
API Name	x_7756_update_set.my_scope_app_sc	Caller Access	-- None --
Client callable		Accessible from	All application scopes
		Active	✓
Description			

Script

```
var my_scope_app_script_include = Class.create();
my_scope_app_script_include.prototype = {
    initialize: function() {},
    execute: function() {
        // write your code here.
    },
    type: 'my_scope_app_script_include'
};
```

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



* 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();

}
```

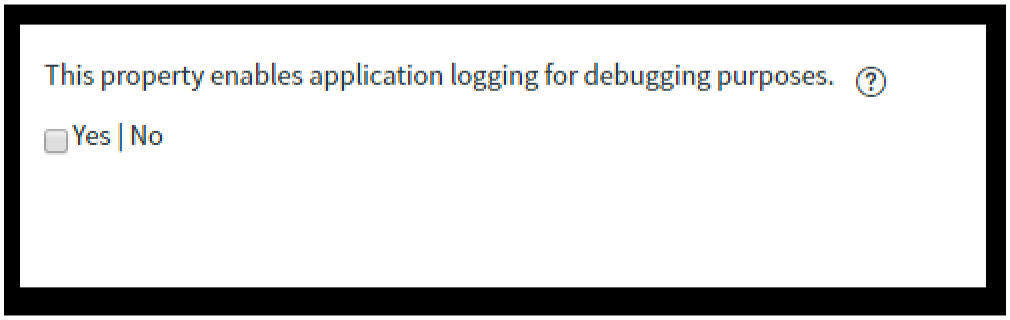
Component Changed

Yes

- Make sure you have a process to cleanup these script 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. Ideal value of 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



This property enables application logging for debugging purposes.

☐ Yes | No

- 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 there are any components that 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 10 minutes, the maximum number of update sets that could be committed is possibly 10 - 12. 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.