

DataSync for ServiceNow Helpful Tips

We've compiled some handy tips and tricks for you to make the most of your Perspectium application. Please explore!

For more basic tips, visit the [Best Practices](#) page.

Assign read-only role to Perspectium for ServiceNow users

The role **perspectium_readonly** can be assigned to any users accessing your ServiceNow instance. Assigning the **perspectium_readonly** role will restrict Perspectium functionality so that users cannot delete, modify, or create new records for the following:

- Shared/Subscribed Queues
- Bulk/Scheduled Bulk/Dynamic Shares
- Table/Transform Maps
- Perspectium Properties

To assign the **perspectium_readonly** role to a user in your ServiceNow instance, follow these steps:

1. Log into your ServiceNow instance as a System Administrator.
2. Using the left-side navigator, go to **User Administration > Users**.
3. Click the name of the user whose role you want to change.
4. Under the **Roles** tab at the bottom of the form, click **Edit**.
5. In the left-side search window under **Collection**, type **perspectium_readonly**.
6. Select the **perspectium_readonly** role and move it to the **Roles** list using the right arrow button (>).
7. Click **Save**.

bulk/dynamic share records too large to send

Records that exceed either ServiceNow or Perspectium's record size limit cannot be bulk or dynamic shared out.

To check which records or fields are too large to send, log into your sharing ServiceNow instance and navigate to **Perspectium > Logs**.

For any records that are too large to bulk/dynamic share out, you will see an error message indicating the record size limit that was exceeded, as well as any fields that were notably large.

Duplicate entries in a table record's activity log

If you have a table, such as **incident**, that has an activity log showing duplicated **sys_journal_field** entries, try de-checking the boxes for **Run business rules** and **Refresh history set** on the incident table subscribe configuration.

Since **Refresh history set** is already checked on the **sys_journal_field** table itself, running refresh history set on the incident table will cause it to refresh twice. Due to the way ServiceNow handles timing and refreshing history set, this can cause two entries to occur.

As well, running business rules can create duplicate entries, since a business rule may cause the system to believe two different events are occurring.

However, turning off **Run business rules** will disable the table events' business rule (for example the **incident events** business rule for the **incident** table) that is run to create events in the event log.

If you want these events to occur, you can execute them in the **Before subscribe** script of the table's subscribe configuration using the ServiceNow function that actually fires the event (**gs.eventQueue()**) and the **gr_before** and **repl_gr** GlideRecord objects available in the **Before subscribe** script.

For example, to cause the **incident.assigned.to.group** event to be fired when the assignment group is changed, you would add the following to the **Before subscribe** script:

```

if (gr_before.assignment_group != repl_gr.assignment_group) {
    gs.eventQueue("incident.assigned.to.group", current, current.assignment_group, previous.assignment_group.
getDisplayValue());
}

```

Note that for fields that are not stored in the table itself and are stored in the **sys_journal_field** table, such as the **comments** field in the **incident** table, you would want to have the script in the **Before subscribe** script of the **sys_journal_field** configuration as follows:

```

if(repl_gr.element == "comments" ){
    var igr = new GlideRecord("incident");
    if(igr.get(repl_gr.element_id)){
        gs.eventQueue("incident.commented", igr, gs.getUserID(), gs.getUserName());
    }
}

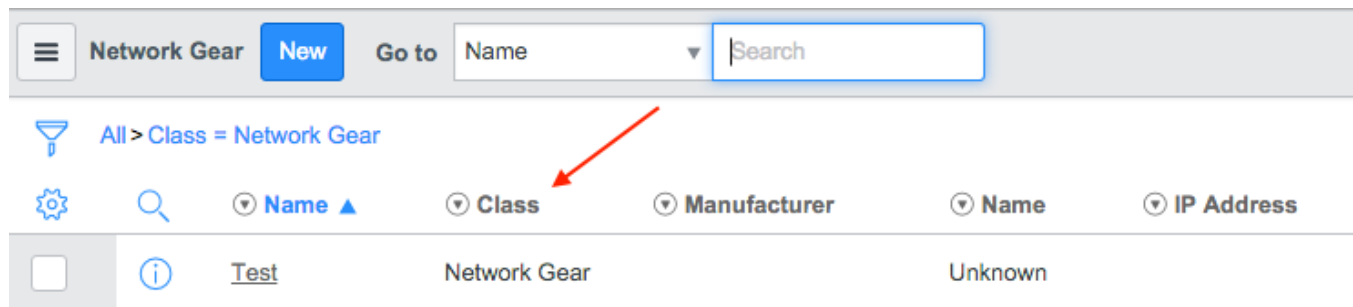
```

In this case, the incoming **sys_journal_field** record is checked to verify that it is a comments record and that is for the incident table by checking if the **sys_id** in the record exists in the incident table. If it does, then the **incident.commented** event is fired using the incident record itself (**igr**) to ensure the event is properly created.

Replicating class name changes between ServiceNow instances

For replicating between ServiceNow instances, changes to the class name are supported so that subscribing instances will also update the record's class name. This is most useful with configuration items (**cmdb_ci**) where discovery runs and changes the class name of configuration items (because they were created with the wrong class name) and you want these changes to replicate properly.

For example, if you have a network gear (**cmdb_ci_netgear**) item:



Network Gear						
		New	Go to	Name	Search	
	All > Class = Network Gear					
			Name		Class	
					Manufacturer	
					Name	
					IP Address	
		<u>Test</u>		Network Gear		Unknown

If you change the class to a different class, such as IP Switch (**cmdb_ci_ip_switch**), Perspectium will send out a **cmdb_ci_ip_switch** record and the subscribing instance will notice the change in class and update it accordingly.



NOTE:

The subscribing instance must be subscribing to all tables that the class name can be changed to (in the above example, if the **cmdb_ci_ip_switch** table were only subscribed to the **cmdb_ci_netgear** table, Perspectium would "skip" the **cmdb_ci_ip_switch** update.

It is recommended you subscribe to global or the base table (such as **cmdb_ci**) in order for class name changes to replicate properly.

Trigger Perspectium from an import set or script

There are occasions when a dynamic share on a table is not getting triggered because the table modification is performed by a script or Import Set Transform Map that is stopping subsequent business rules from running. In the latter case, the **Run business rules** checkbox may have been unselected. In the other case, the **setWorkflow(false)** API may have been called.

In either case, you can trigger Perspectium directly by inserting the following code snippet in your script at the right position.

NOTE: You must first create a dynamic share configuration for the table.

```
var psp = new PerspectiumReplicator();
psp.shareRecord(GR, "table_name", "operation");
```

Where:

- GR = the current GlideRecord
- "table_name" = the table name of the GlideRecord
- "operation" = the mode you want to trigger replication - the options are "insert", "update", or "delete"

Dot walking field values to be replicated

On occasion, you may need to access a field's value using ServiceNow's [Dot Walking feature](#).

The following example leverages this feature by sending the display value of a sys_domain's *name* value, subscribing it to update another instance of Servicenow.

Place the following line in the **Before Share** script of your bulk shares or dynamic shares:

```
current.sys_domain = current.sys_domain.name;
```

This will modify the outgoing record's payload to have the domain's name in the **sys_domain** column.

If you are subscribing this into another ServiceNow instance, you must handle it either with a Transform Map, or, you could set up a similar Before Subscribe script to mirror this by grabbing the Domain corresponding to this name, like so:

```
var domainVal = repl_gr.sys_domain;
if(domainVal != null){
    var dGR = new GlideRecord('sys_user_group');
    dGR.addQuery('name', domainVal);
    dGR.queryNoDomain();
    if(dGR.next()){
        current.sys_domain = dgr.sys_id;
    }
}
```

Adjusting date/time for local time zones before replication

Date/Time fields in ServiceNow are stored in the database in UTC timezone. They are adjusted for the individual user's local timezone as defined by their profile at runtime in the UI. This allows anyone viewing the data to see date/time values in their local timezone to avoid confusion. When we replicate that data, we just replicate it as is in UTC, and write it to the target, without doing any kind of timezone offset, since there isn't one in the context of a machine integration. Typically reporting solutions can account for this and adjust based on your end user's needs.

This is fairly standard across most enterprise applications.

If you want to explicitly convert all data to a specific timezone for replication you can do so using a **BeforeShare** script in bulk shares and dynamic shares and specify the fields that you want to convert.



We do not recommend this, as it can cause issues if the reporting or viewing technology being used adjusts it again in their UI. You also need to consider the impact of Daylight Savings. Something converted and replicated during Standard Time could be off by an hour compared to something converted during Daylight Savings time.

If you wish to do this anyways, here is a simple example script which converts **sys_updated_on** and **opened_at** to US/Eastern timezone during replication.

```
// Date/Time variables you want to update
var timesToUpdate = ["opened_at", "sys_updated_on"];
var curTimeZone = "America/New_York";

// Get the specified timezone
var tz = Packages.java.util.TimeZone.getTimeZone(curTimeZone);

// Edit specified variables with the offset
var time;
var timeZoneOffset;
for(var t in timesToUpdate){
    time = new GlideDateTime(current.getValue(timesToUpdate[t]));
    time.setTZ(tz);
    timeZoneOffset = time.getTZOffset();
    time.setNumericValue(time.getNumericValue() + timeZoneOffset);
    current.setValue(timesToUpdate[t], time);
}
```

Ignore or cancel share

There are three ways you can control a dynamic share to only fire on certain field updates:

Manually, in a BeforeShare script

Ignoring or canceling a share in the before share script

In the **Before Share** script of a dynamic or bulk share configuration, you can set the global variable **ignore** to the boolean value **true** to prevent the current record from being shared.

For example, the following script ignores the dynamic sharing of an **incident** record when the **priority** field value is **1**:

```
if (current.priority == 1) {
    ignore = true;
}
```

As another example, the following script will ignore sharing the record with a **number** value **TKT0010001** during bulk sharing of all **ticket** records:

```
if (current.number == "TKT0010001") {
    ignore = true;
}
```

Only trigger when specified columns have changed

Ignoring a share if only one field has changed

For cases where you have a table's records updated frequently but data doesn't actually change (such as a table that gets updated every single day via another integration or ServiceNow Discovery), you may not want the table's dynamic share (with interactive only not selected) to run and share out any records.

As an example, consider the field that gets updated every day is **u_last_discovered_date**. The rest of the fields don't usually change, and you don't want to share these records out again since the subscribing side (such as a database) doesn't really need the latest **u_last_discovered_date**.

In these cases, you can run the following script to ignore sharing the record:

```
function listChangedFields(obj){
    var flds = [];
    var aud = new GlideRecord("sys_audit");
    aud.addQuery("documentkey", obj.sys_id);
    aud.addQuery("record_checkpoint", obj.sys_mod_count);
    aud.query();
    while (aud.next()){
        flds.push(aud.getValue("fieldname"));
    }
    return flds;
}

var changedFields = listChangedFields(current);
var ignoreFields = ["priority", "urgency"]; // If any changed field falls outside that list, the update will
be sent

ignore = true;

var util = new ArrayUtil();
for (var i=0; i<changedFields.length; i++){
    if (!util.contains(ignoreFields, changedFields[i])) ignore = false;
}
```

Ignoring a share with multiple field changes

In the **Additional Settings** tab of the table, check the **Select column updates to ignore** box to ignore sharing records with multiple field changes. This will reveal the related list, allowing you to select the desired fields that you want to be ignored when updated.

Trigger Conditions

Additional Settings

Filter and Enrichment

Scheduled Sync Up

Include journal fields

☐

Include audit log

☐

Include attachments

☐

Include embedded images/videos

☐

Include referenced field records

☐

Select column updates to ignore

☒

Conditional Share

☐

Table map

Q

View name

Target queue

Q

Update

Delete

Related Links

[Activate all](#)

[Deactivate all](#)

[Reset Dynamic Share Rule](#)

[View message set activity](#)

Log messages (4)

PSP Conditional Shares

Updated columns to ignores (2)

Updated columns to ignores

New

Go to

Column Name

Search

1 to 2 of 2

Dyname Share = incident

Column Name

☐

i

number

☐

i

description

Actions on selected rows...

1 to 2 of 2


Considering the picture above as an example, the record would be ignored if the **number** and **description** fields are the ONLY fields that have been updated; if any other fields have also been updated, the record will not be ignored.

Only trigger when columns other than the specified columns have changed

Sharing on specific field changes

In the **Filter and Enrichment** tab of the table, check the **Select column updates to share on** box to share a record only when one of any number of chosen fields are updated. This will reveal the related list, allowing you to select the desired fields.

Trigger Conditions Additional Settings **Filter and Enrichment** Scheduled Sync Up Notes

Select column updates to share on ☒ 

Share only selected fields ☐

Condition Add Filter Condition Add "OR" Clause

-- choose field -- -- oper -- -- value --

Before share script

After share script

Note that you are only able to select one option out of **Select column updates to share on** and **Select column updates to ignore**. Checking either box will cause the other one to disappear.

Next, select the fields that you want to trigger a share. Considering the picture below as an example, the record would **ONLY** be shared if the **assigned_to** or **description** fields have been updated; if these fields have not been updated, the record will be ignored.

Related Links

[Activate all](#)
[Deactivate all](#)

Log messages (2) PSP Conditional Shares Updated Columns to ignore or Sh... Bulk Shares

Updated Columns to ignore or Share Ons New Go to Column Name Search

Dynamic Share = ticket

Column Name

☐ assigned_to

☐ description

Actions on selected rows...

Sharing out HTML fields

For tables that have HTML fields, such as the **kb_knowledge** table and its **text** field, use the **encrypted_multibyte** encryption mode to ensure that the HTML fields are sent out properly. Otherwise, HTML fields may be sent with extraneous characters for spaces, as seen below:

Two Words with a single space between them.

Two Words with two spaces between them.

Sentences with single space. Following each period.

Sentences with two spaces. Following each period.

Carriage Returns: 1) Empty, 2) Empty, 3) Single Space, 3) Double Space

┆

┆

┆

┆

End carriage Returns

┆

By default, ServiceNow instances and the DataSync Agent support the various encryption modes out of the box, so there is no additional configuration required on the subscribing side.

Multiple MultiOutput jobs

If it seems like you are not sending data out of your instance as fast as you would like, ask these questions

- Is the count of my Outbound Messages [psp_out_message] consistently very high (+250k Ready Message)?
- Is my property for maximum bytes per post too low (should be in the 5 MB to 10 MB range)?
- Is my property for max record per post too low (should be around 2000-4000 records)?
- How often is my Perspective MultiOutput Processing job running (default to 30 seconds)?

These are the typical things we look at for optimizations first prior to adding in Multiple MultiOutput jobs. If these are all set as expected and the throughput is still not enough, learn more about [multiple jobs here](#).

Slow subscribe performance

If you notice slow subscribe performance in your instance, check **System Diagnostics > Slow Queries** to see if any slow queries are running on subscribe. See if these slow queries are a result of **Run business rules** selected on the subscribe, and turn those off if not needed.

If you see a field that a query is always running on, see if that field is indexed or not, and if not, add an index to that field to help improve performance.

Display scroll bar in UI pages

A scrollbar is visibly shown for ui pages with multiple modules on them. This feature works for the Google Chrome browser and Windows machine users. If you are using Mac OS, you must configure your system preferences to visibly display the scrollbar automatically.

Send archive records as delete actions using bulk configuration

1. Create a bulk share for the archive record, i.e. **Table name** is ar_incident. Then, select **Share the table records only** in the **Sharing setup** dropdown.
2. In the bulk share, navigate to **Runtime Setting > Runtime Behavior**. Then, enable **Share as archive**.
3. Navigate to **Scripting > Script Options**. In the **Before share script** field, add the following line of code to force the delete action:

```
psp_action = "delete";
```

4. Fill the rest of the required fields, i.e. Target Queue, etc.
5. **Save** and **Execute** the bulk share and check to confirm the outbound message is sent correctly on the original archive table, i.e. incident.delete.

Add Indexes to Perspectium system tables

The Perspectium outbound and inbound message queues in ServiceNow in particular requires indexes to be added in order for DataSync to function optimally.

The following indexes are already included in the Core update set:

Table	Index Name	Type	Fields
PSP Out Message tables (psp_out_message, u_psp_attachment_out_message, u_psp_audit_out_message, u_psp_observer_out_message)	psp_out_query	composite, non-unique	State (state) Target Queue (u_target_queue) Created (sys_created_on)
PSP Out Message tables (psp_out_message, u_psp_attachment_out_message, u_psp_audit_out_message, u_psp_observer_out_message)	psp_out_query3	single, non-unique	Sequence (u_sequence)
PSP Out Message tables (psp_out_message, u_psp_attachment_out_message, u_psp_audit_out_message, u_psp_observer_out_message)	psp_out_query4	composite, non-unique	Created (sys_created_on) Sequence (u_sequence)
PSP In Message (psp_in_message)	psp_in_query	single, non-unique	Created (sys_created_on), name, key, u_sequence
PSP In Message (psp_in_message)	psp_in_query2	single, non-unique	State (state)
PSP Log Message (u_psp_log_message)	psp_log_query	single, non-unique	Created (sys_created_on)
PSP Log Message (u_psp_log_message)	psp_log_query2	composite, non-unique	Created (sys_created_on) Type (u_type)
PSP Properties (u_psp_properties)	u_psp_properties_u_name	single, non-unique	Name (u_name)
PSP Replicate Conf (u_psp_replicate_conf)	u_psp_replicate_conf	single, non-unique	active, sync_direction, table_name

If these indexes are not included, follow [Create a table index](#) or contact [Perspectium Support](#).