Error Handling

Errors in Exalate

Exalate has a built-in error recovery mechanism that allows you to fix problems and resume synchronization from the moment it fails.

Synchronization can fail due to a variety of causes, such as:

- Network connectivity
- Authentication issues
- Configuration failures
- Compatibility problems
- Sync Rules/Script errors

When an error arises in Exalate, synchronization is paused to allow an administrator to rectify the problem and resume operations.

Depending on the type of error, the synchronization is paused at one of these error levels:

- Entity level
- Connection level
- Exalate Application level

For instance, when Exalate detects a script error, the connection is paused. All other synchronizations, which are not related to that particular Connection, can proceed as usual.

If there is an error such as a network connectivity problem, all Connections are paused until the connectivity problem is fixed. Exalate resolves network errors automatically. This means that the synchronization will be recovered automatically once the network connectivity problem is resolved.

If the Exalate App has a problem, the whole synchronization will be stopped.

Errors Page

In the Exalate admin menu, you can access the **Errors** tab to get an overview of all the errors.

Here you will see a list of the existing errors. An error overview includes an error summary description, its impact level, and the time when the error occurred.

You can also get to the **Error Details** page from here:

Error	Impact	Occurred At	Action
Issue Tracker Error Rest exception while scheduling sync events for issues for connection 'hp_to_jira'. Please check if you have domain `DEFAULT` and project `...	All Sync is blocked	2020-03-22 14:14:34	•••
			Details
			Resolve and Retry

Error Details

Error details include the following information:

- **Impact**: The title (Issue, Relation, Warning, Node, Trigger) indicates the type of error occurring in the backend. This may impact either a certain issue or a connection synchronization. It may also affect the whole instance.
- **Connection**: The link to the Connection in which the error has occurred.
- **Error Type: T**he type of error.
- **Error Detail Message**: Provides a detailed description of what's wrong.
- **Error Stack Trace:** Helps to find the root cause of an error.

Back to Errors

- Impact:

  ◆ ISSUE

- Local Issue:

- Remote Issue:

  RED-1

- Connection:

  ✖ cloud-red

- Error Type:

  TrackerRestException

- Error Creation Time:

  2018-04-11 08:11:31

- Error Detail Message:

  Rest exception while creating an issue with json body: {"fields":{"summary":"test","description":"new description info","reporter":{"name":"admin"}}}
 Response status code: 400
 Response body: {"errorMessages":[],"errors":{"project":"project is required"}}

- Error Stack Trace:

  com.exalate.replication.services.api.issuetracker.rest.TrackerRestException: Rest exception while creating an issue with json body: {"fields": {"summary":"test","description":"new description info","reporter":{"name":"admin"}}}
 Response status code: 400
 Response body: {"errorMessages": [],"errors":{"project":"project is required"}} at services.jcloud.transport.RestErrorHandlingService$.filterErrorResponse(RestErrorHandlingService.scala:11) at services.jcloud.transport..JCloudClient$$anonfun$5.apply(JCloudClient.scala:274) at services.jcloud.transport..JCloudClient$$anonfun$5.apply(JCloudClient.scala:274) at scala.util.Success$$anonfunmap1.apply(Try.scala:236) at scala.util.Try$.apply(Try.scala:191) at scala.util.Success.map(Try.scala:236) at services.jcloud.transport..JCloudClient.createIssue(JCloudClient.scala:274) at services.jcloud.hubobjects.JCloudHubObjectService.createIssue(JCloudHubObjectService.scala:140) at com.exalate.replication.services.processor.CreateIssueProcessor$$anonfun$createIssue$4.apply(CreateIssueProcessor.scala:73) at com.exalate.replication.services.processor.CreateIssueProcessor$$anonfun$createIssue$4.apply(CreateIssueProcessor.scala:71) at scala.concurrent.Future$$anonfun$flatMap$1.apply(Future.scala:251) at scala.concurrent.Future$$anonfun$flatMap$1.apply(Future.scala:249) at scala.concurrent.impl.CallbackRunnable.run(Promise.scala:32) at scala.concurrent.impl.ExecutionContextImpl$AdaptedForkJoinTask.exec(ExecutionContextImpl.scala:121) at scala.concurrent.forkjoin.ForkJoinTask.doExec(ForkJoinTask.scala:260) at scala.concurrent.forkjoin.ForkJoinPool$WorkQueue.runTask(ForkJoinPool.java:1339) at scala.concurrent.forkjoin.ForkJoinPool.runWorker(ForkJoinPool.java:1979) at scala.concurrent.forkjoin.ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:107)

To fix the script, you can access the Connection from the **Error Details** page.

- Impact:

  ◆ ISSUE

- Local Issue:

- Remote Issue:

  RED-1

- Connection:

  ✖ cloud-red

- Error Type:

  TrackerRestException

- Error Creation Time:

  2018-04-11 08:11:31

- Error Detail Message:

  Rest exception while creating an issue with json body: {"fields":{"summary":"test","description":"new description info","reporter":{"name":"admin"}}}
 Response status code: 400
 Response body: {"errorMessages":[],"errors":{"project":"project is required"}}

- Error Stack Trace:

At the very bottom, you can also check the local and remote replica values. They are displayed as a prettified *JSON* file.

Error Detail Message:

Processor exception detected during processing of the remote 'TEST-961' using connection 'cloud_to_8' javax.script.ScriptException: java.lang.RuntimeException: Create processor error !

Error Stack Trace:

com.exalate.api.processor.ProcessorException: Processor exception detected during processing of the remote 'TEST-961' using connection 'cloud_to_8' javax.script.ScriptException: java.lang.RuntimeException: Create processor error ! at com.exalate.replication.request.CreateIssueSyncRequestState.transition(CreateIssueSyncRequestState.java:101) at com.exalate.replication.request.CreateIssueSyncRequestState.transition(CreateIssueSyncRequestState.java:28) at com.exalate.replication.in.RequestProcessorService.processSyncRequest(RequestProcessorService.java:312) at com.exalate.replication.in.RequestProcessorService.processSyncRequestsForIssue(RequestProcessorService.java:169) at com.exalate.replication.in.RequestProcessorService.processSyncRequests(RequestProcessorService.java:114) at com.exalate.replication.in.RequestWorker$1.run(RequestWorker.java:81) at com.exalate.util.ClusteredSensitiveExecutorService.executeHandlingLocks(ClusteredSensitiveExecutorService.java:32) at com.exalate.replication.in.RequestWorker.run(RequestWorker.java:74) at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511) at java.util.concurrent.FutureTask.runAndReset(FutureTask.java:308) at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$301(ScheduledThreadPoolExecutor.java:180) at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:294) at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) at java.lang.Thread.run(Thread.java:748) Caused by: com.exalate.api.processor.ProcessorException: javax.script.ScriptException: java.lang.RuntimeException: Create processor error ! at com.exalate.processor.ExalateProcessor.executeProcessor(ExalateProcessor.java:48) at com.exalate.processor.ExalateProcessor.executeProcessor(ExalateProcessor.java:33) at com.exalate.processor.jira.JiraCreateIssueProcessor.executeCreateProcessor(JiraCreateIssueProcessor.java:188) at com.exalate.processor.jira.JiraCreateIssueProcessor.createIssue(JiraCreateIssueProcessor.java:116) at com.exalate.replication.request.CreateIssueSyncRequestState.transition(CreateIssueSyncRequestState.java:79) ... 14 more Caused by: javax.script.ScriptException: javax.script.ScriptException: java.lang.RuntimeException: Create processor error ! at org.codehaus.groovy.jsr223.GroovyScriptEngineImpl.eval(GroovyScriptEngineImpl.java:151) at javax.script.AbstractScriptEngine.eval(AbstractScriptEngine.java:264) at com.exalate.processor.ExalateProcessor.execute(ExalateProcessor.java:76) at com.exalate.processor.ExalateProcessor.executeProcessor(ExalateProcessor.java:40) ... 18 more Caused by: javax.script.ScriptException: org.codehaus.groovy.jsr223.GroovyScriptEngineImpl.eval(GroovyScriptEngineImpl.java:348) at org.codehaus.groovy.jsr223.GroovyScriptEngineImpl.eval(GroovyScriptEngineImpl.java:145) ... 21 more Caused by: java.lang.RuntimeException: Create processor error ! at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method) at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62) at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45) at java.lang.reflect.Constructor.newInstance(Constructor.java:423) at org.codehaus.groovy.reflection.CachedConstructor.invoke(CachedConstructor.java:77) at org.codehaus.groovy.runtime.callsite.ConstructorSite$ConstructorSiteNoUnwrapNoCoerce.callConstructor(ConstructorSite.java:102) at org.codehaus.groovy.runtime.callsite.CallSiteArray.defaultCallConstructor(CallSiteArray.java:57) at org.codehaus.groovy.runtime.callsite.AbstractCallSite.callConstructor(AbstractCallSite.java:182) at org.codehaus.groovy.runtime.callsite.AbstractCallSite.callConstructor(AbstractCallSite.java:190) at Script130.run(Script130.groovy:1) at org.codehaus.groovy.jsr223.GroovyScriptEngineImpl.eval(GroovyScriptEngineImpl.java:345) ... 22 more

Copy to clipboard   Show local replica   Show remote replica

Below you can see how the replica values look.

Back

```
{
    "version": {
        "major": 1,
        "minor": 15,
        "patch": 0
    },
    "hubIssue": {
        "key": "TEST-961",
        "summary": "test error level #2",
        "project": {
            "id": 10000,
            "key": "TEST",
            "name": "TEST"
        },
        "labels": [],
        "reporter": {
            "key": "557058:ada77a72-56a9-4a19-9c23-7d66e4813883",
            "active": true,
            "email": "serhii@idalko.com",
            "displayName": "Serhii Syrotskyi",
            "username": "admin"
        },
        "attachments": [],
        "comments": [],
        "status": {
            "id": "10000",
            "name": "To Do",
            "description": ""
        },
        "issueType": {
            "id": "10104",
            "name": "Bug"
        },
        "priority": {
            "id": "3",
            "name": "Medium"
        },
        "entityProperties": {},
        "customKeys": {}
    }
}
```

Resolve and Retry

Once the error is fixed, you can check to make sure the fix has worked and proceed with the synchronization.

Use the **Resolve and Retry** button accessible from the error list to do that.

If the error persists, generate a support.zip file and contact support.

Decrease Error Impact

Exalate allows you to decrease the error impact level.

If the error does not affect all the entities under sync, you may change the error impact to the entity level. The synchronization will be blocked only for this specific entity.

The example below shows how the impact column looks on the error list screen:

Ignore Related Sync

Exalate allows you to ignore errors related to the sync.

If the error is ignored, changes related to the sync will not be applied. If it is the first sync, new entities will not be created.

While the **Ignore Related Sync** operation can be helpful for dealing with temporary errors, it's essential to exercise caution. Not all errors should be ignored. Some errors might be related to critical data discrepancies or configuration problems that need immediate attention.

How to Set up Error Notifications?

In Exalate, you can notify administrators in case an error is raised.

When an error arises, a notification will be sent out to either the **exalate-administrators** group (if configured) or to the **Jira-administrators** group.

There are 3 types of notifications:

- The in-app notification brings up a pop-up window.
- An email notification containing the error message.
- A sync status highlighting that something went wrong.

Note: Exalate can send error notifications only if SMTP is configured on your server.

> **Note**: Exalate sends error notifications in Jira on-premise and Jira Cloud once an hour.

Notifications on Jira on-premise

Types of Notifications

1. **Email Notifications**

   Administrator(s) will receive notifications each time there is an error in the synchronization process.

   You can choose which users should be informed of an error.
   When an error is raised, a notification will be sent out to either the **exalate-administrators** user group (if configured) or to the **jira-administrators** group.

   You can configure a new group of **exalate-administrators** with the users who should be notified in case of an error.

   For more information, check out How to configure error notifications in Jira on-premise.

2. **In-JIRA Notifications** will result in a pop-up notification, which would be shown to admins until the error is resolved or the notification is closed.
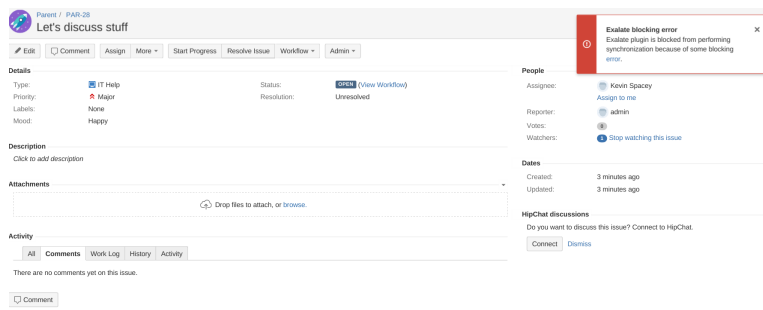
Configuring Notifications on Jira

Navigate to **Jira > Admin > Add-ons > Exalate > General Settings** and enable desired notification type.
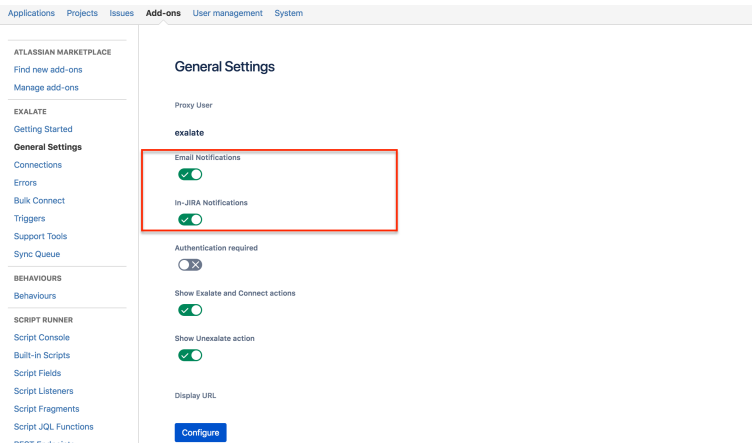
Notifications on Other Platforms

Types of notifications

1. **Email Notifications**

   Administrator(s) will receive notifications each time there is an error in the synchronization process. You can add users who should be informed of an error.
   When an error is raised, a notification will be sent out to the users, specified in the Exalate admin menu under the **Exalate Notifications** tab.
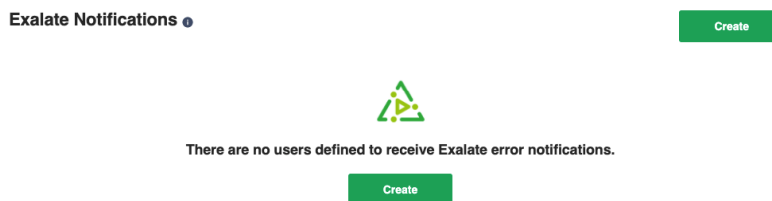
2. **In-App Notifications** will result in a pop-up notification, which would be shown to admins until the error is resolved or the notification is closed.

   Below you can see the pop-up notification.
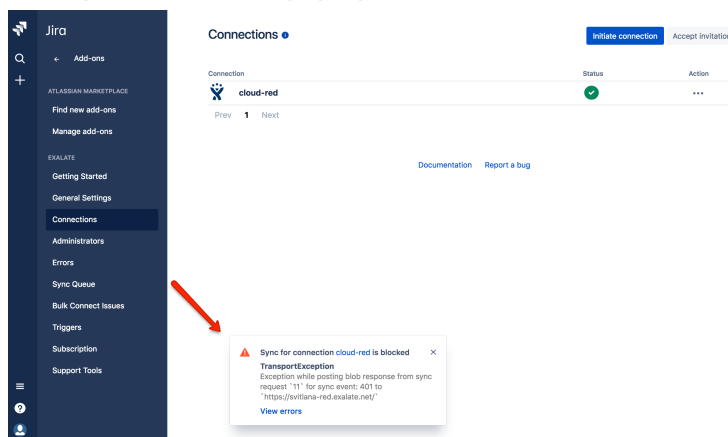
Configuring Notifications on Other Platforms

In-app notifications are displayed by default.

Email notifications could be configured in the Exalate admin menu. Navigate to the **Exalate admin menu> Exalate Notifications** and add an administrator to receive email notifications.
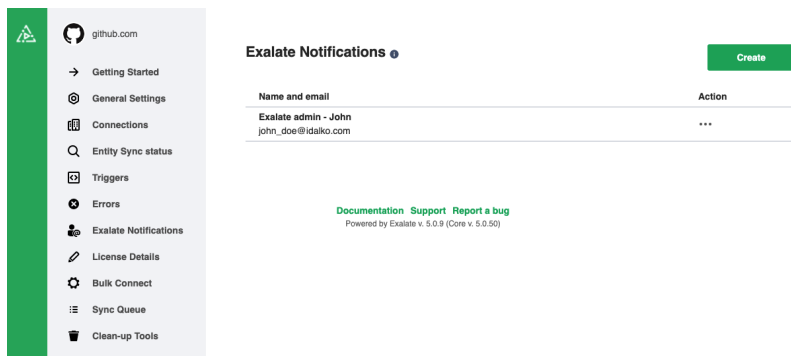
Error Debugging

Sometimes you need to have a closer look at synchronization scripts to find out what's wrong.

There are two ways to get feedback from the Exalate App:

- Exalate logging: it doesn't block the synchronization and requires the processor changes to run again. We discuss this in the next section.
- Debugging error from the scripts - blocks the entity synchronization and requires **Resolve and retry** to run again.

Debugging Errors from the Scripts

To get more details about values passed in a replica you can use **debug.error** method.

debug.error method

This method helps to raise an IssueTrackerException error with the value of the field which is not syncing.

It allows you to see the content of the replica and the issue (if applicable). This helps to get a better understanding of what stage the problem has occurred.

```
debug.error("String message")
```

For example, when you synchronize a custom field, but the issue on the receiving side is not updated and you don't see any error.

The script below shows how to throw an error with the custom field value stored in a replica.

Receiving side - Incoming sync

```
debug.error("customFields."Foo".value from remote side is: " + replica.customFields."Foo"?.value)
```

How to Get Only Debug Logging for the Scripts?

If you want to debug your scripts, then capturing all **com.exalate** logging is too much as it also captures all transport protocol details.

You can limit the logging to different levels:

- for only script logging use **com.exalate.script**
- for information on how scripts run use **com.exalate.processor**

For Exalate 4.2 and lower

- for scripts and information on how they run use **com.exalate.processor**
- for Outgoing sync(data filter) scripts

  use **com.exalate.processor.jira.JiraCreateReplicaProcessor**
- for Incoming sync for new issues(create processor) scripts

  use **com.exalate.processor.jira.JiraCreateIssueProcessor**
- for Incoming sync for existing(change processor) scripts use

  com.exalate.processor.jira.JiraChangeIssueProcessor

You can test how logging works by adding the code, provided below. It will add the 'Hello world' text to the log file.

```
//add a 'Hello world' to the logging
log.info("Hello world ...")
```

You can also enable debug (temporary and permanent) logging in Jira on-premise.

Error Logging

You can enable logging for the Exalate App. For example in Jira on-premise, the log file is stored in <JIRA-HOME>/log directory.

It doesn't block the synchronization and requires the processor changes to run again.

> **Note**: With the updated 2.17.2 version of Log4j, there were some changes to the way logs are generated. Prior to this version, the Exalate log file was generated as **exalate.log**. With the new version, Exalate logs are now generated by default in the **atlassian-jira.log** file, instead of the previous location as a separate exalate.log. However, if needed you can still generate a separate file as before by using your Jira advanced configuration settings.

Troubleshooting the Errors

You have an error and you want to find the root cause of that error. We recommend that you follow some basic troubleshooting steps:

1. Check the **Error Details** to see the stack trace behind the error.

The **Error Details** page helps to get more details about the cause of an error.
2. Check if the error impact can be decreased.

Exalate allows you to **decrease the error impact** in case the error affects only a specific issue. This will unblock the rest of the synchronization. So that you can troubleshoot this specific issue.

Check how to change the error impact for more details.
3. Resolve and Retry to resume the synchronization from where it was paused.

Once the error is fixed, use the **Resolve and Retry** button to proceed with the synchronization.

Why isn't a particular Field Syncing?

You don't have an error anymore but you realize that the field you want to send over to the destination instance isn't syncing.

This can be because:

- The field is not being sent.
- The field is being sent but is not applied properly on the receiving side.

Solution

To check if the field is being sent:

- Make sure it is included in the Outgoing sync of the sending instance
- If you are using Jira Server, check for its presence on the remote issue tab.
- If you have an error, check the remote replica (available on the Error details)

To check if the field is properly applied on the receiving side:

- Make sure you are mapping it in the Incoming sync.
- For fields where there is a set of available options (resolution, priority, list custom fields, user fields) make sure you have the proper option on the receiving instance.
- You can throw errors for not unexpected cases to validate the behavior in an easier way.

An example of the Resolution field

1. Check if the Sync Rules have been configured properly. You can find more details in our guide on how to synchronize resolution in Jira on-premise and Jira Cloud.
2. If it didn't help, add the script below to the Incoming sync. This script helps to check whether the issue is caused by different Resolutions used on the remote side.
   It will raise an error if the Resolution is not found on your side.

```
if (replica.resolution == null || nodeHelper.getResolution(replica.resolution?.name) == null) {
  throw new com.exalate.api.exception.IssueTrackerException("Failed to find resolution: `${replica.resolution?.name}
`".toString())
}
```

Steps to Follow when Exalate Node is Unavailable

Whenever the Exalate node can't be accessed by a user there is a set of recommended steps to follow:

1. Check out the Exalate node status at status.exalate.com

   Here you can see the current node status as well as Scheduled Maintenance.

2. Check out status.cloud.google.com

   The Exalate status page doesn't always hold a reference to original incidents from Google cloud.

   If Exalate service is down it's worthwhile to check Google's own status.

3. Contact our support team

   If you can't find any information on the scheduled maintenance but still cannot access the Exalate node, please contact our support team for further troubleshooting.

   You can submit a support request to our Service Desk directly. Or just send an e-mail to support@exalate.com. This will create an issue on our JIRA Service Desk allowing for further tracking.

Have more questions? Ask the community