

How to Update Violations Metadata

The built-in Static Analysis Prioritization example is designed to teach you how to write a flow that retrieves static analysis violations from DTP through the DTP REST API and update their metadata in bulk. The flow is sectioned into several examples that serve as discussion points for learning how flows and nodes function. In this tutorial you'll learn how to set the Static Analysis Prioritization example flow up and how individual processes in the flow execute specific functions.

The following DTP REST APIs are used in this example:

- GET /v1.2/staticAnalysisViolations
- POST /v1.4/staticAnalysisViolations/metadata
- GET /v1.4/metadata/priority

Table of Contents:

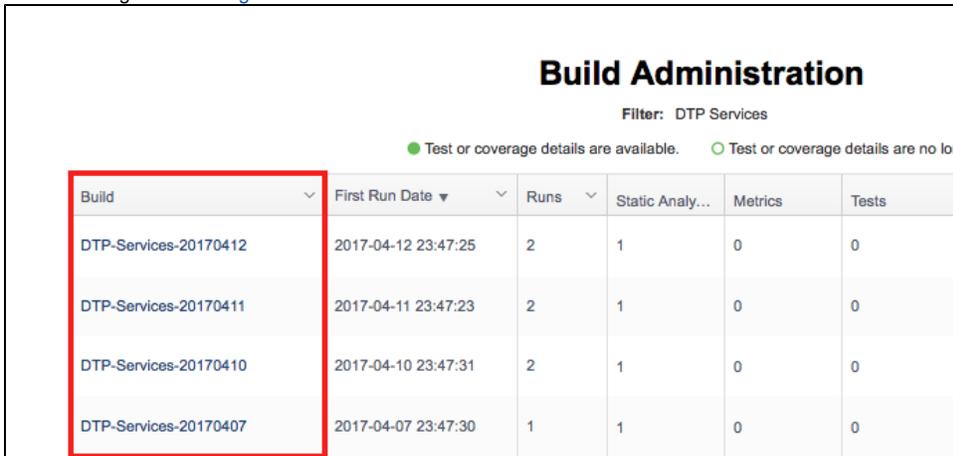
- [Requirements](#)
- [Example 1 - Set Assignee](#)
- [Example 2 - Set Priority](#)
- [Example 3 - Set Action](#)
- [Example 4 - Set Risk/Impact](#)
- [Example 5 - Set Due Date](#)
- [Example 6 - Set Reference Number](#)
- [Example 7 - Add a Comment](#)
- [Example 8 - Prioritize by Severity](#)

Requirements

- DTP and Extension Designer must be configured correctly. See [Server Settings](#).
- DTP should have collected static analysis data from a Parasoft code analysis and test execution tool (i.e., C/C++test, dotTEST, or Jtest).

Setting Up the Example

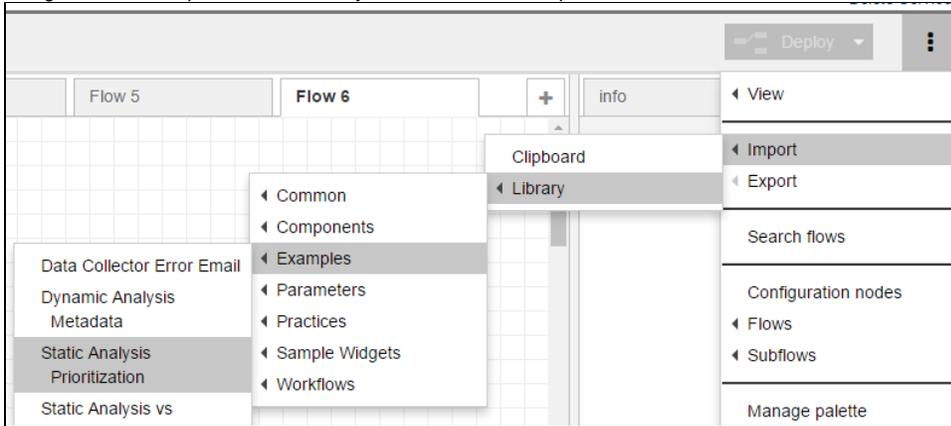
1. Get the build ID for a static analysis execution from the Build Administration page in DTP. This page is accessible from the [Build Administration - Statistics](#) widget. See [Using Build Administration](#) for additional information.



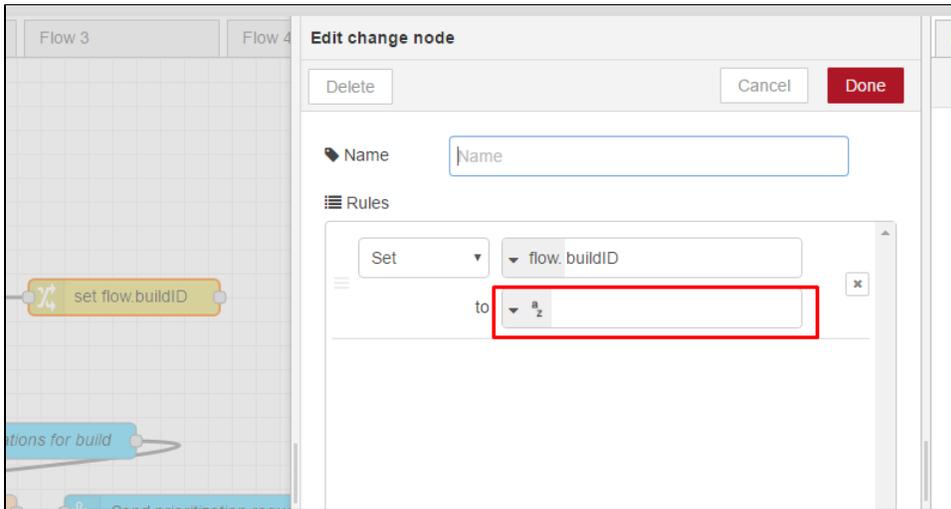
Build	First Run Date	Runs	Static Analy...	Metrics	Tests
DTP-Services-20170412	2017-04-12 23:47:25	2	1	0	0
DTP-Services-20170411	2017-04-11 23:47:23	2	1	0	0
DTP-Services-20170410	2017-04-10 23:47:31	2	1	0	0
DTP-Services-20170407	2017-04-07 23:47:30	1	1	0	0

2. Click on an existing service or create a new service in Extension Designer (see [Working with Services](#)).

3. Create a new flow (see [Working with Flows](#)) and choose **Import > Library > Examples > Static Analysis Prioritization** from the Extension Designer menu to import the Static Analysis Prioritization example.

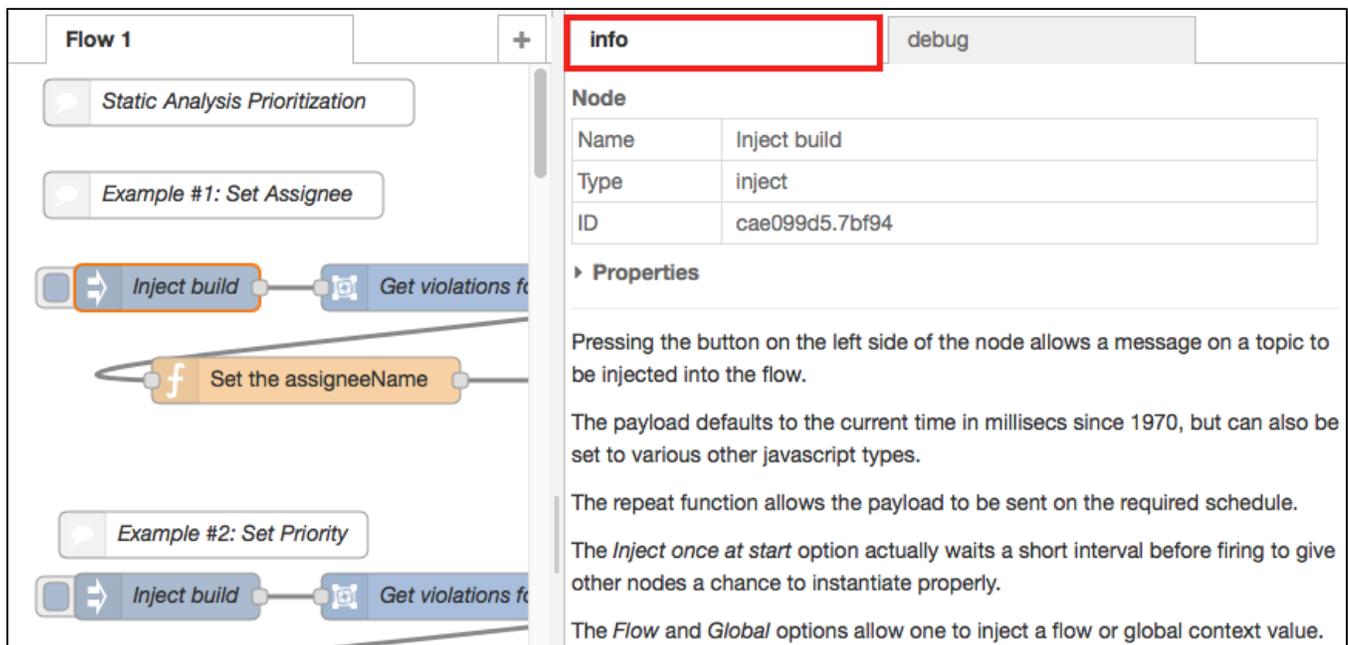


4. Click **Deploy** and double click the `set flow.buildID` node.
5. Enter the build ID value in the **to** field and click **Done**.



6. Deploy the updated flow to finish preparing the example.

If you need additional help understanding how a node works, you can click on the node to view its documentation in the Info tab.



Example 1 - Set Assignee

The nodes in Example #1 provide a method for assigner all retrieved violations to a new user.

Retrieving Static Analysis Violations from DTP

The flow retrieves static analysis data via the /staticAnalysisViolation REST API provided by DTP. See [Using DTP APIs](#) for details.

/staticAnalysisViolations

[Collapse List](#) | [More Information](#)

GET /staticAnalysisViolations [Returns static analysis violations](#)

Description

Returns a list of static analysis violations, filtered by the search criteria if provided. The violations must match all criteria: that is, the search performs a logical AND of the criteria.

Double-click the **Get violations for Build** node. This node is a DTP REST API node, which is a specific type of node for making requests to the DTP REST API (see [Working with Nodes](#) for all available nodes). The endpoint in the node points to /v1.2/staticAnalysisViolations?buildId={{payload}}. This node retrieves data based on the build ID specified when you [set up the flow](#). It returns the static analysis violations payload to msg.staticAnalysis.

The screenshot shows the 'Edit DTP REST API node' dialog box. The fields are as follows:

- Name: Get violations for build
- Method: GET (default)
- Endpoint: /v1.2/staticAnalysisViolations?buildId={{payload}}
- Response: msg.staticAnalysis

The background shows a flow editor with a node labeled 'Get violations for build' highlighted in red.

Processing Data from DTP to Set New Assignee

Double-click the **Set the assigneeName** function node. The node contains JavaScript that processes the violations and prepares for a new payload from the next REST API call.

The screenshot shows a Node-RED flow editor with a function node named "Set the assigneeName" highlighted in red. The "Edit function node" dialog is open, displaying the following JavaScript code:

```

1 var _ = context.global.lodash;
2
3 // note: splitting violations by 100. Sending large data at once to DTP REST API
4 // an error and unable to perform correctly.
5 var violationsArrayby100 = _.chunk(msg.staticAnalysis.violations, 100);
6
7 var prioritize = [];
8 _.forEach(violationsArrayby100, function(violations) {
9   prioritize.push({
10    metadata: {
11      ids: _.map(violations, 'id'),
12      applyToAllBranches: true,
13      changes: { assignee: 'johndoe' },
14    }
15  });
16 });
17
18 // returning prioritization violation 100 a time by putting into array of array
19 //[[msg, msg, msg]]
20 return [prioritize];
21

```

Extension Designer functions use the lodash JavaScript utility library. The first part of the code initializes lodash as a variable, which enables you to sort and split arrays of violations data.

```
var _ = context.global.lodash;
```

Line 5 splits the `staticAnalysisViolations` array into multiple arrays of 100 violations each. This reduces the payload to a safe size when sending back to DTP REST API. Processing a large payload in a single call may trigger an unexpected error from Extension Designer (see [Working with Nodes](#)).

Lines 7 through 16 create new msg arrays to prepare for the next prioritization API call. This API expects a JSON object. Each example will show you different way to set fields under Violations metadata. Review API documentation to learn more about the acceptable payload.

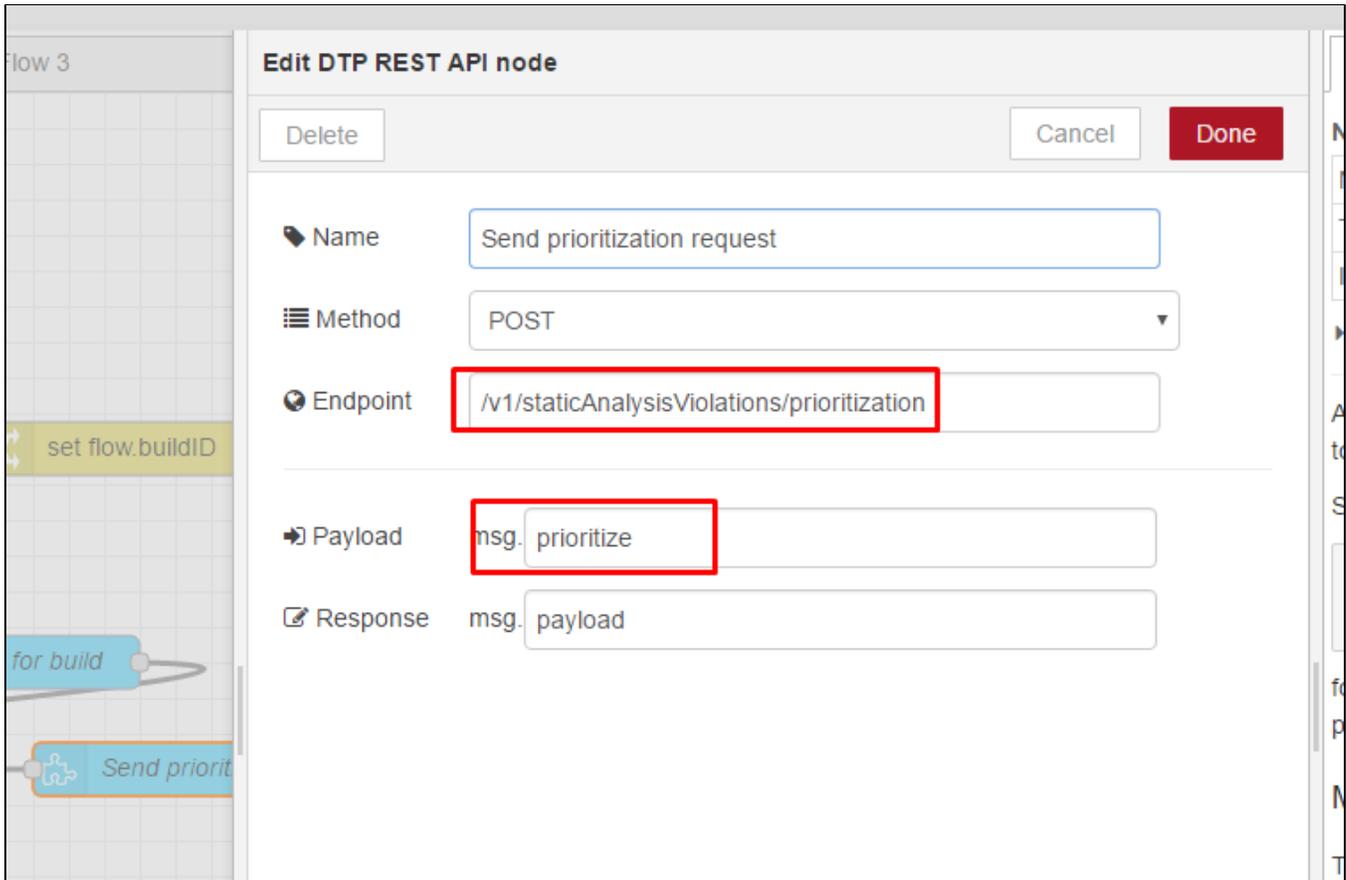
POST	/staticAnalysisViolations/metadata	Updates metadata for violations
Description		
Updates the metadata (prioritization data) for a given set of violations, and/or adds a comment to the history of a set of violations.		
Returns the set of modified violations. A violation is considered modified if the value of one or more of its metadata fields was changed (the field's previous value was not equal to the value in the request); or if a comment was added to the violation. If a comment is included in the request, then all of the given violations are considered modified because a comment is added to each violation.		
The field "referenceNumber" accepts any string. The field "dueDate" accepts a date string. For valid "priority", "classification", and "violationAction" values, see the name values provided by the GET /metadata/{entityType} endpoint. Only the fields "referenceNumber" and "dueDate" can be set to null.		
The "applyToAllBranches" parameter is optional: the default value is false.		

In this example, the assignee is set to johndoe. All violations called in the API will be assigned to johndoe once the flow is processed. You can add your own logic to assign different users if necessary.

The flow will only be able to assign violations to existing DTP users who are members of the project associated with the data. See [User Administration](#) for details.

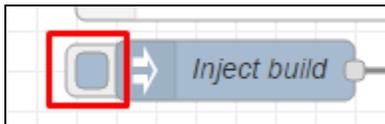
At line 20, the flow sends a two dimensional array of one element. You can also send multiple messages in a specific order. This function node only expects one output by sending an array of message objects into the output. The flow will send msg objects one at a time to the next flow, which is similar to looping through the array and returning each message asynchronously (see <https://nodered.org/docs/writing-functions#multiple-messages> for additional information).

The payload is prepared for the prioritization API. The next DTP REST API node completes the transaction.



The prioritization API is an unpublished, undocumented, and unsupported API. However, the API allows you to POST with a prioritize payload and apply changes to all violation IDs passed to it.

Click the **Inject Build** node to run the flow and see outputs in the debug tab.



Based on number of violations, you may see multiple transactions from the prioritization REST API.



Example 2 - Set Priority

All logic flows in this example differ only in terms of preparation of the Prioritization payload.

Double-click the **Find and set the priorityId** function node.

Name 

Function

```
1 var _ = context.global.lodash;
2
3 // note splitting violations by 100. Sending large data at once to DTP REST A
4 // an error and unable to perform correctly.
5 var violationsArrayby100 = _.chunk(msg.staticAnalysis.violations, 100);
6
7 var prioritize = [];
8 _._forEach(violationsArrayby100, function(violations) {
9     prioritize.push(
10      {
11        metadata: {
12          ids: _._map(violations, 'id'),
13          applyToAllBranches: true,
14          changes: {
15            priority: 'Critical'
16          }
17        }
18      }
19    );
20 });
21
22 // returning prioritization violation 100 a time by putting into array of arr
23 //[[msg, msg, msg]]
24 return [prioritize];
25
```

Lines 10 through 18 contain the JavaScript for setting the priority. In the example, priority is set to Critical for all violations passed.

Example 3 - Set Action

Double-click the **Set the violationAction** function node.

Name

Set the violationAction



Function

```
1 var _ = context.global.lodash;
2
3 // note splitting violations by 100. Sending large data at once to DTP REST API
4 // an error and unable to perform correctly.
5 var violationsArrayby100 = _.chunk(msg.staticAnalysis.violations, 100);
6
7 var prioritize = [];
8 _._forEach(violationsArrayby100, function(violations) {
9   prioritize.push(
10    {
11     metadata: {
12       ids: _.map(violations, 'id'),
13       applyToAllBranches: true,
14       changes: { violationAction: 'Fix' }
15    }
16  });
17 });
18 });
19
20 // returning prioritization violation 100 a time by putting into array of array of
21 // [[msg, msg, msg]]
22 return [prioritize];
23
```

Lines 11 through 15 contain the JavaScript for defining what action should be taken on the violations. In this example, `violationAction` is set to `Fix`.

Actions are defined in DTP. You can query the `/v1.4/metadata/violationAction` API to see what values are available for the `violationAction` field. You can use any available value specified in the response.

Request URL

```
http://skunk-dev4.parasoft.com:8080/grs/api/v1.4/metadata/violationAction
```

Response

```
Access-Control-Allow-Methods: GET, POST, OPTIONS  
Content-Type: application/json;charset=UTF-8  
Access-Control-Allow-Origin: *  
Access-Control-Allow-Headers: Content-Type, Authorization
```

```
{  
  "count": 6,  
  "entities": [  
    {  
      "id": 2,  
      "name": "Fix"  
    },  
    {  
      "id": 1,  
      "name": "None"  
    },  
    {  
      "id": 6,  
      "name": "Other"  
    },  
    {  
      "id": 3,  
      "name": "Reassign"  
    },  
    {  
      "id": 4,  
      "name": "Review"  
    },  
    {  
      "id": 5,  
      "name": "Suppress"  
    }  
  ]  
}
```

This metadata can be customized and every DTP server may have different set of actions. Review the `/metadata/violationAction` API documentation for additional information.

Example 4 - Set Risk/Impact

This example is very similar to [Example 2 - Set Priority](#) and [Example 3 - Set Action](#). In this example, all violations are classified as Extreme. Double-click the **Find and set the classificationId** function node and review the JavaScript.

Example 5 - Set Due Date

Double-click the **Set the dueDate** function node and review the JavaScript. The Moment JavaScript library is initialized in line 2. Moment is used in this example to get the date and time in a string. Moment is also used to calculate tomorrow's date as a string to set `dueDate` field.

Name

Function

```
1 var _ = context.global.lodash;
2 var moment = context.global.moment;
3
4 // note splitting violations by 100. Sending large data at once to DTP REST API
5 // an error and unable to perform correctly.
6 var violationsArrayby100 = _.chunk(msg.staticAnalysis.violations, 100);
7
8 var prioritize = [];
9 _._forEach(violationsArrayby100, function(violations) {
10     prioritize.push(
11         {
12             metadata: {
13                 ids: _.map(violations, 'id'),
14                 applyToAllBranches: true,
15                 changes: { dueDate: moment().add(1, 'days').format('YYYY-MM-DD')
16             }
17         }
18     );
19 });
20
21 // returning prioritization violation 100 a time by putting into array of array
22 //[[msg, msg, msg]]
23 return [prioritize];
24
```

4/13
msg.
ol
e

4/13
msg.
{
4/13
msg.
{
4/13

Outputs 1

Example 6 - Set Reference Number

Double-click the **Set the referenceNumber** function node and review the JavaScript. In this example, the `referenceNumber` field is set in the violation metadata.

Edit function node

Delete Cancel **Done**

Name 

Function

```
1 var _ = context.global.lodash;
2 // note splitting violations by 100. Sending large data at once to DTP E
3 // an error and unable to perform correctly.
4 var violationsArrayby100 = _.chunk(msg.staticAnalysis.violations, 100);
5
6 var prioritize = [];
7 // loop through violationsArrayby100 and construct prioritize message of
8 _._forEach(violationsArrayby100, function(violations) {
9     prioritize.push(
10        {
11            metadata: {
12                ids: _.map(violations, 'id'),
13                applyToAllBranches: true,
14                changes: { referenceNumber: '12345' }
15            }
16        }
17    );
18 });
19
20 // returning prioritization violation 100 a time by putting into array c
21 //[[msg, msg, msg]]
22 return [prioritize];
23
```

The reference number could be used for the ID of external systems, such as JIRA and Bugzilla, for later use.

Example 7 - Add a Comment

Double-click the **Set the comment** function node and review the JavaScript. In this example, a comment is added to the violation.

Edit function node

Delete
Cancel
Done

Name 📄 ▼

Function

```

1  var _ = context.global.lodash;
2
3  // note splitting violations by 100. Sending large data at once to DTP E
4  // an error and unable to perform correctly.
5  var violationsArrayby100 = _.chunk(msg.staticAnalysis.violations, 100);
6
7  var prioritize = [];
8  // loop through violationsArrayBy100 and construct prioritize message of
9  _.forEach(violationsArrayby100, function(violations) {
10     prioritize.push(
11         {
12             prioritize: {
13                 fields: {},
14                 applyToAllBranches: true,
15                 comment: 'Comment added using Service Designer',
16                 violationIds: _.map(violations, 'id')
17             }
18         }
19     );
20 });
21
22 // returning prioritization violation 100 a time by putting into array c
23 //[[msg, msg, msg]]
24 return [prioritize];
25

```

Notice that comments are not part of the `fields` object. A comment is an additional property associated with the `prioritize` object.

Example 8 - Prioritize by Severity

This is an advanced example that demonstrates how you can group static analysis violations based on severity and process the violations accordingly.

Double click the **Group violations and set relevant priorityId** function node and review the JavaScript. Instead of returning an array from the `msg` array, the `node.send()` method is used to push a `msg` object to the next node asynchronously (see <https://nodered.org/docs/writing-functions#sending-messages-asynchronously> for additional information about sending messages asynchronously).

Edit function node

Delete Cancel Done

Name

Function

```

1 var _ = context.global.lodash;
2 var groups = _.groupBy(msg.staticAnalysis.violations, 'severity');
3
4 _._each(groups, function(violations, severity) {
5   var violationsby100 = _.chunk(violations, 100);
6   _._each(violationsby100, function(vios) {
7     node.send({
8       metadata: {
9         ids: _._map(vios, 'id'),
10        applyToAllBranches: true,
11        changes: { priority: _._find(msg.priority.entities, {"id": parseInt(severity)}).name }
12      }
13    });
14  });
15 });
16

```

You can query the `/v1.4/metadata` DTP REST API to see the metadata values available in your DTP.

/metadata Collapse List | More Information

GET `/metadata/{entityType}` Returns metadata entities

Description
Returns a list of all metadata entities of the selected type.

Try It Out
This will send a request to the Development Testing Platform REST API.

Request Parameters

Parameter	Value	Data Type	Description
entityType	<input type="text" value="priority"/> <ul style="list-style-type: none"> priority classification riskImpact testAction violationAction 	string	The type of metadata entity to retrieve.

Response Defin Sample JSON | JSON Schema