COAR Notify - LDN Services

- Relation with the Quality Assurance Correction Service
- LDN Autodiscovery mechanism
- LDN Services Directory (Registering Services)
- LDN Inbox queueing
- Notify status boxes
- Configuring automatic QA evaluation using the Level of Trust
- Sending LDN Notifications during the submission of an item
- Automatic Inbound pattern triggering
- Item filters for Inbound pattern
- The LDN Consumer
- Understanding the structure of the LDN Notification
  - @context
  - actor
  - context
  - id
  - inReplyTo
  - object
  - origin
  - target
  - type
- POSTMAN COLLECTION

Relation with the Quality Assurance Correction Service

The LDN system, as a message exchanging i/o system, has an inbox and an outbox. Every LDN message refers to a Notify Service: all the Notify Services are configured manually from the admin application form. A Notify Service is just like an authority labelled on LDN messages.

The Quality Assurance system is the implementation of item metadata updates operations. A Quality Assurance Event contains information for item metadata updates: QAEvent are stored into QAEvent solr collection. All of the QAEvent are shown on an administration form. Every QAEvent can be accepted, ignored or removed: if accepted some metadata of the referred item are modified, if ignored or removed nothing about the item is modified. Every QAEvent has a property named source: qa events created by processing an LDNMessage has source="coar-notify".

To process an LDN message means to create a QAEvents; as soon as the QAEvent is accepted the referred item is updated. We do it by routing LDN Messages from the queue to the LDN Router. The LDN Router is designed into the spring file on the bean ldn. The router de-multiplex two attributes read from the LDN Message (two string values inside the array called type) content to java classes called Processors: see org.dspace.app.ldn.processor.* package to view them all.

The match between LDN message type and the QAEvent topic (suddenly created by the evaluation of the ldn message from the queue) is configured onto ldn-coar-notify.xml spring file. Every Processor owns a list of actions. An action is often an email to be sent and an action to create the qa event. The LDN processor receives the QAEvent topic as a parameter and creates the relative QAEvent.

LDN Autodiscovery mechanism

Third party system can retrieve the location of the repository LDN InBox via the LDN autodiscovery mechanism, nevertheless to be able to interact with DSpace they need to be approved by a Repository Administrators and listed in the LDN Services Directory (see next paragraph); otherwise their messages will be flagged as untrusted and not processed at all.

LDN Services Directory (Registering Services)

You need to register the external services to allow the handling of the incoming LDN messages.

These services are also used to send LDN notification during the submission process of an item (Please take a look at LDN Inbound pattern)
Administrators can manage services using the menu **LDN Services**.

The following page is used to submit a new LDN Service:

**Create service**

**Name**

Please provide service name

**Description**

Please provide a description regarding your service

**Service URL**

Please input the URL for users to check out more information about the service

**Level of trust**

Please enter a value between 0 and 1. Use the **.”** as decimal separator

**Service IP range**

IPv4 range lower bound

IPv4 range upper bound

Please enter a valid IPv4 in both range bounds. Note: for single IP, please enter the same value in both fields

**LDN Index URL**

Please specify the URL of the LDN Index

**Inbound Pattern**

**Item Filter**

**Automatic**

- **Send**

**Name**: a label for the Notify Service used on the UI

**Description**: the description of the Notify Service - add details here
**Service URL:** the url of the remote Notify Service. This is mostly used as a descriptive URL when sending notification to an external system.

The service URL must not be confused with the inbox url. As said this url is descriptive so we expect the main application URL to be added here.

**Level of Trust:** floating point number value accepted between 0 and 1.

This value is used to assign to the service a TRUST value that describes how much reliable the external service is.

This value is not used if the automatic processing of QA Events is disabled

Check the configuration file dspace/config/spring/api/qaevents.xml to enable the automatic approval

When the automatic approval is configured the trust value of the LDN Service is check against configured score thresholds.

Thresholds for automatic Approval/Rejection/Ignoring can be set.

Also within a specified range QA Events will need a manual approval

**Service IP Range:** two IP addresses expected as minimum and maximum.

This range is used to validate the received LDN Messages.

When the LDN notification doesn't match the configured IP Range the notification is stored as UNTRUSTED

**LDN Inbox URL:** this is the url used to send the LDN Notifications.

This url is also used when a notification is received to retrieve the registered LDN service it belongs to

This URL must is unique among the registered LDN Services

**Inbound Pattern:** the section for Inbound pattern is the section which describes what operations/actions are supported by the external service

The pattern itself can be selected from the dropdown as there's a list of pattern. To better understand pattern usage please refer to the official documentation https://notify.coar-repositories.org/patterns/

The item filter is used to describe which item can be processed for the current LDN Service/Inbound pattern.

If the item filter is not set any item is allowed. If the filter is selected only for matching items the LDN Notification will be sent (in case of automatic LDN Notification the notification is not sent. In case of NON-automatic service a validation will prevent requests to be sent to the external service)

The automatic flag when set to true the ldn message exchange is performed automatically right after the item submission has finished. This Automatic workflow generates an Outgoing LDN message targeting the Notify service for the just submitted item.

The automatic flag involves only the submission phase of an item.

If no item filter is set - the LDN notification is generated for any submitted item.

---

**LDN Inbox queueing**

LDN incoming messages are stored into the ldn_message database table. As far as the property ldn.enabled is true and the incoming json is valid, the LDN Message is stored on the table. Together with the storage of the record, the queue status of the message is initialized. The queue_status column of the table contains the status of the LDN message inside the queue. All the possible queue_status values are described into the java class org.dspace.app.Ldn.LDNMessageEntity as integer constants.

<table>
<thead>
<tr>
<th>Status name</th>
<th>Value in DB</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUEUE_STATUS_UNTRUSTED IP</td>
<td>0 5</td>
<td>Message must not be routed as it is not trusted. This may occur if the IP address of the notifications' sender doesn't match the provided &quot;IP Range&quot; or if the service inbox URL in the origin section of the message doesn't match with any registered LDN Service entry's Inbox URL.</td>
</tr>
<tr>
<td>QUEUE_STATUS_QUEUED</td>
<td>1</td>
<td>Message is waiting in the queue to be processed by the Extractor</td>
</tr>
<tr>
<td>QUEUE_STATUS_PROCESSING</td>
<td>2</td>
<td>Message is currently being processed by the Extractor</td>
</tr>
<tr>
<td>QUEUE_STATUS_PROCESSED</td>
<td>3</td>
<td>Message has been <strong>successfully</strong> processed by the Extractor</td>
</tr>
<tr>
<td>QUEUE_STATUS_FAILED</td>
<td>4</td>
<td>Message has been evaluated but its routing has failed</td>
</tr>
</tbody>
</table>
If all validations are run successfully the LDN Notification status is set as **QUEUE_STATUS_QUEUED** and the notification will be processed as soon as the extractor retrieves it from the queue.

The LDN Message logical queue is managed by:

- **The LDN Message Extractor** is an asynchronous DSpace task which retrieves the oldest processable LDN message. A LDN notification is processable by the Extractor only if its queue status is **QUEUE_STATUS_QUEUED** other status will not be considered by the extractor. The extractor process ends as soon as the LDN extracted message is routed and processed (either with success or failure).

- **The LDN Message Timeout Checker** is an asynchronous DSpace task that looks for timed-out messages with attempts less than Y, where Y= the value of the configuration property `ldn.processor.max.attempts`. Cron configuration at property `ldn.queue.timeout.checker.cron` Each notification if not successfully processed is retrieved in the next execution of the Checker and attempts are kept up to date (increased by 1 and the timeout is increased by X minutes), where X= the value of the configuration property `ldn.processor.queue.msg.timeout` | defaulted to 60.

Please consider that this means that the corresponding QAEvent is not automatically created as soon as the LDN Message is received. The QA Event related to the LDN Notification will be created only once the notification is successfully processed by the extractor (LDN queue status set to **QUEUE_STATUS_PROCESSED**).

**Notify status boxes**

Considering these possible scenarios here at: [COAR Notify Protocol: Example Scenarios](#)

We have to keep the user updated about the item situation. We do it with colored boxes on its landing (handle) page.

When the Offer message (being it review, endorsement or ingest) has been sent as an outgoing LDN message, and nothing else about it has been received, the yellow box is shown.

When the Offer message has been followed by a related incoming Acknowledgement message: if the ack is a tentative reject the box shown is red.

When Offer message has been followed by a related Announce incoming message, the box shown is blue! It is blue because receiving an Announce means - then the message is extracted - we produce a new QA Event. As every QA Event it is shown on Item Page and myDSpace page.

**Configuring automatic QA evaluation using the Level of Trust**
Score is a number $0 < \# < 1$.
If the proper configuration is enabled an automatic check for approval of QAEvents is run once the LDN message is extracted.
On qaevents.xml file the bean qaScoreEvaluation we can configure three different boundaries to automatically approve/reject/ignore the level of trust:

```xml
<property name="scoreToReject" value="0.3" />
<property name="scoreToIgnore" value="0.5" />
<property name="scoreToApprove" value="0.8" />
```

- if score $\leq$ rejection deletes the QAEvent;
- if score $\leq$ ignore discards the QAEvent;
- if score $>$ approval accepts the QAEvent automatically
- if score respects the following: $\mathrm{scoreToIgnore} < \mathrm{score} < \mathrm{scoreToApprove}$
  no automatic action is performed and the user must check manually the QA event

This feature must not be confused with the automatic triggering for LDN Inbound pattern

### Sending LDN Notifications during the submission of an item

In the example below it is shown the submission of an Item with a new section for COAR-Notify.
In this section we can manually choose ldn-services (which were not marked as automatic) in order to ask for Review, Endorsement and Ingest to external services.
If No (non-automatic) pattern has been configured for the LDN Service this service will not be displayed in any of the dropdown.

![COAR Notify](image)

**Automatic Inbound pattern triggering**

When adding new LDN Inbound pattern to a LDN Service we can select if a pattern has to be automatic or not.
Flagging a pattern as automatic means that the request described by the current pattern is sent automatically to the external service:

- No action is required during the submission to select the LDN Service for a specific pattern
- The service in not even listed in the COAR notify step of the submission

**Item filters for Inbound pattern**

When adding inbound pattern to a service we have the possibility to add an item-filter for each described pattern

An item filter allows to filter items according to specified criteria.

- If the pattern is flagged as automatic the item filter is evaluated automatically
  - if the item filter is respected the LDN Notification is stored and queued for sending
  - if the item filter is not respected the LDN Notification is not even generated an nothing will be sent to the external service
- If the pattern is not flagged as automatic the item filter is evaluated when choosing the service from the dropdown for Notify in the submission
  - if the pattern is not respected an error message will be displayed saying the selected LDN service cannot be used for the current Item
  - if the pattern is respected no error message is displayed
Item Filter is a non-mandatory field. It activates a filter to be applied to the item during the submission. If the item matches the item filter condition the submission will be successful; otherwise the submission will fail with a talkative message. Visible item filters are configured into a static list mapped into item-filters.xml file - on a bean named ldnItemFilters. We established for the following item filter to be shown:

- **Has one Bitstream** = the item must have exactly one bundle named "ORIGINAL";
- **Item is public** = anonymous user can see/read the item;
- **Title starts with pattern** = the dc.title metadata of the item must start with "Pattern";
- **Type equals dataset** = the dc.type metadata of the item must be 'Dataset';
- **Type equals Journal Article** = the dc.type metadata of the item must be 'Journal Article';

current item-filter.xml has all item filters declarations: only the ones declared inside the ldnItemFilters map are shown.

The LDN Consumer

The LDN Consumer is the consume responsible for generating the **outgoing** LDN Notification. This consumer is triggered any time an Item is successfully installed and a check against the registered LDN Services/patterns is done to generate the appropriate LDN Notification. This consumer is responsible for the handling of both automatic a non-automatic patterns

```
event.consumer.ldnmessage.class = org.dspace.app.ldn.LDNMessageConsumer
event.consumer.ldnmessage.filters = Item+Install
```

Understanding the structure of the LDN Notification

The LDN Notification includes some important section many of these if improperly set might led to errors in evaluations on the notification. Let's take a deeper look at the LDN Notification structure

@context

The context section is the same for any LDN Notification
actor
This section describes the actor which performs the request
This is a descriptive section and doesn't include important information

```
"actor": {
  "id": "https://review-service.com",
  "name": "Review Service",
  "type": "Service"
}
```

context
This is one of the most important section.
In this section we have a description of the Item being involved in the LDN Notification

- **id** is the identifier URL of the item in the repository
  this field is really important to identify and link the Item to the current notification

other fields are descriptive fields for the item

```
"context": {
  "id": "https://research-organisation.org/repository/preprint/201203/421/",
  "ietf:cid-as": "https://doi.org/10.5555/12345680",
  "type": "sorg:AboutPage",
  "ietf:item": {
    "id": "https://research-organisation.org/repository/preprint/201203/421/content.pdf",
    "mediaType": "application/pdf",
    "type": [
      "Article",
      "sorg:ScholarlyArticle"
    ]
  }
}
```

id
The identifier of the LDN notification (Each notification has a different id)

```
"id": "urn:uuid:94ecae35-dcfd-4182-8550-22c7164fe23f"
```

inReplyTo
This field is used to reply to other LDN notification so that the system knows if the current notification is a response

```
"inReplyTo": "urn:uuid:0370c0fb-bb78-4a9b-87f5-bed307a509dd"
```

object
This is a descriptive section of the received Review, Endorsement etc..

- **id** is the URL of the review/endorsement etc.
origin
This is an important section since it is used (for incoming notifications) to determine the service among the registered ones. If the service is not recognized as registered the notification is untrusted.

The field being used for this check is inbox

target
This section is used to describe the target system involved in the notification workflow

type
This type section is important since it's determining how to process/evaluate any received notification

POSTMAN COLLECTION