1. Introduction

The Domino mail service is now included in the extension library. The mail service represents mail views in JSON format and mail messages in both JSON and MIME format. The mail service lets you send HTTP requests to:

- Discover a list of mail resources for the authenticated user.
- Read the list of messages in the inbox, sent view and drafts view.
- Delete a message.
- Send a message in either JSON or MIME format.
- Save and update a draft message in either JSON or MIME format.

The mail service is the latest addition to the family of REST services collectively called Domino Access Services. Like the Domino data service, the mail service needs to be enabled by a Domino administrator. If you are familiar with the data service, you should find the mail service easy to work with. It shares some of the same URL parameters and other conventions used by the data service.

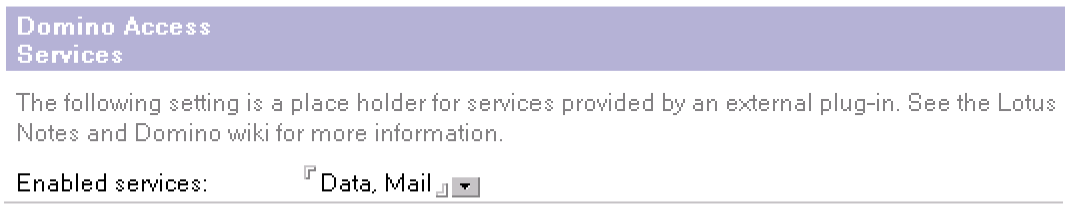
The following sections help you get started with the mail service. As the mail service evolves, we will publish a more complete reference document.

2. Enabling the Mail Service

After you install the extension library, the mail service is loaded whenever the Domino HTTP task is started. However, an administrator typically doesn't want the mail service to handle requests on every Domino server. You need to deliberately enable the mail service in the appropriate Internet Site document.

To enable the mail service:

1. Use a Notes client to open the server's public address book.
2. In the Domino Directory navigator, select Configuration – Web – Internet Sites.
3. Open the Internet Site document for your server.
4. Click the Edit Web Site action.
5. Select the Configuration tab.
6. At the bottom of the form, look for a section labeled Domino Access Services.
7. In the Enabled services field, add the Mail keyword:


Domino Access Services

The following setting is a place holder for services provided by an external plug-in. See the Lotus Notes and Domino wiki for more information.

Enabled services: Data, Mail

8. Save your changes and restart the HTTP task.

NOTE: The above instructions assume you are using Internet Sites. If you are not using Internet Sites, you can enable the mail service in the server document. See the Domino Data Service documentation for more information.

3. Developer Guide

3.1 Discovering Mail Resources

To get the list of resources for the authenticated user, you send an HTTP GET request to the service's root resource:

GET /api/mail

The mail service returns a response in JSON format like this:

```
{
  "mailboxes":[
    {
      "owner":{
        "displayName":"Duke Lawson",
        "distinguishedName":"Duke Lawson\/Peaks",
        "email":"DukeLawson@swg.usma.ibm.com"
      },
      "links":[
        {
          "rel":"inbox",
          "href":"\/mail\/dlawson.nsf\/api\/mail\/inbox"
        },
        {
          "rel":"outbox",
          "href":"\/mail\/dlawson.nsf\/api\/mail\/outbox"
        },
        {
          "rel":"sent",
          "href":"\/mail\/dlawson.nsf\/api\/mail\/sent"
        },
        {
          "rel":"drafts",
          "href":"\/mail\/dlawson.nsf\/api\/mail\/drafts"
        },
        {
          "rel":"trash",
          "href":"\/mail\/dlawson.nsf\/api\/mail\/trash"
        }
      ]
    }
  ]
}
```

In other words, the response is an object containing a **mailboxes** array. Each mailbox object represents a mail file to which the authenticated user has access. Each mailbox has the following properties:

- **owner** is an object describing the owner of the mail file.
- **links** is an array of resources corresponding to the mail file. There are links for the inbox, outbox (used to send mail), sent view and drafts view.

NOTE: When sending a GET request to /api/mail, you must include credentials identifying the user. These can be basic or session authentication credentials. If the credentials are either missing or invalid, the mail service will return an authentication challenge (e.g. HTTP 401 in the case of basic auth).

3.2 Reading the Inbox

To get the list of messages in the inbox, you send an HTTP GET request to the inbox resource URI:

GET /{database}/api/mail/inbox

where {database} is a place holder for the actual mail file name. Ideally you should get the exact URI from the list of mail service resources described in section 3.1. As an example, the following request reads the inbox in /mail/dlawson.nsf:

GET /mail/dlawson.nsf/api/mail/inbox

The mail service returns a response in JSON format like this:

```
[
  {
    "href":"\/mail\/dlawson.nsf\/api\/mail\/messages\/10C667B7FFD62F61852579A6006864B1",
    "from": {
      "displayName":"Raymond Chan"
    },
    "subject":"Address test -- outgoing mail from Raymond",
    "date":"2012-02-16T19:01:01Z"
  },
  {
    "href":"\/mail\/dlawson.nsf\/api\/mail\/messages\/3DECEA4C1D796261852579A600682F2D",
    "from": {
      "displayName":"Dean Melnyk"
    },
    "subject":"Address test -- outgoing mail from Deak",
    "date":"2012-02-16T18:59:15Z"
  }
]
```

The response is an array of objects. Each object summarizes a single message. For example:

- **href** is the unique URI for the message.
- **from** is an object representing the sender of the message.
- **date** corresponds to the date the message was received.

When reading the inbox you can also specify parameters to read a page at a time, sort the results, search for specific messages, or find messages by key. For example, consider the following request:

GET /{database}/api/mail/inbox?ps=50&page=9

This specifies a page size of 50 messages (**ps=50**) and requests the 10th page in the inbox (**page=9** because page numbers are zero-based).

Now consider this request:

GET /{database}/api/mail/inbox?sortcolumn=date&sortorder=descending

This changes the sort order to be descending by date.

For more information on request parameters see the data service documentation at:

http://www-10.lotus.com/ldd/ddwiki.nsf/dx/Viewfolder_entries_GET_dds10

The mail service supports the following parameters: ps. page, start, si, count, sortcolumn, sortorder, search, searcchmaxdocs, keys and keysexactmatch. The mail service also includes the Content-Range header in responses as described in the data service documentation.

3.3 Reading a Single Message

To read a single message, you send a GET request to the message resource URI:

GET /{database}/api/mail/messages/{unique-id}

The mail service responds with a representation of the message in JSON format:

```
{
  "from": {
    "displayName":"Raymond Chan",
    "distinguishedName":"Raymond Chan\/Peaks"
  },
  "to": [
    {
      "displayName":"Duke Lawson",
      "distinguishedName":"Duke Lawson\/Peaks@Peaks",
      "email":"DukeLawson@swg.usma.ibm.com"
    },
    {
      "displayName":"Dean Melnyk",
      "distinguishedName":"Dean Melnyk\/Peaks@Peaks",
      "email":"DeanMelnyk@swg.usma.ibm.com"
    }
  ],
  "subject":"Address test -- outgoing mail from Raymond",
  "date":"2012-02-16T19:01:01Z",
  "messageId":"<OF10C667B7.FFD62F61-ON852579A6.006864B1-852579A6.006876D9@LocalDomain>",
  "href":"\/mail\/dlawson.nsf\/api\/mail\/messages\/10C667B7FFD62F61852579A6006864B1",
  "content": [
    {
      "contentType":"multipart\/alternative;
Boundary=\"0__=0ABBF335DFFBE2218f9e8a93df938690918c0ABBF335DFFBE221\"",
      "contentDisposition":"inline"
    },
    {
      "contentType":"text\/plain; charset=US-ASCII",
      "data":"\r\n\r\n\r\ntest",
      "boundary":"--0__=0ABBF335DFFBE2218f9e8a93df938690918c0ABBF335DFFBE221"
    },
    {
      "contentType":"text\/html; charset=US-ASCII",
      "contentDisposition":"inline",
      "data":"<html><body>\r\n<p><font size=\"2\" face=\"sans-serif\">test<\/font><\/body><\/html>",
      "boundary":"--0__=0ABBF335DFFBE2218f9e8a93df938690918c0ABBF335DFFBE221"
    }
  ]
}
```

Table 3.3.1 lists the currently supported JSON properties for a message.

Table 3.3.1 JSON properties for a message

Property	Type	Description
from	Object	An object representing the sender of the message. This object may contain one or more of the follwing string properties: displayName, distinguishedName, and email.
to	Array of objects	An array of objects – one for each recipient in the Notes SendTo item.
cc	Array of objects	An array of objects – one for each recipient in the Notes CopyTo item.
bcc	Array of objects	An array of objects – one for each recipient in the Notes BlindCopyTo item.

Property	Type	Description
subject	String	The subject of the message.
date	Date	The date the message was sent.
messageId	String	The message ID (corresponds to the Notes $MessageID item).
inReplyTo	String	The ID of the parent message (corresponds to the Notes In_Reply_To item).
href	String	The URL of this message.
content	Array of objects	An array of parts – similar to a multipart message in MIME format.

You can also choose to read a message in MIME format. To do this, you add **format=mime** to the request URI:

GET /{database}/api/mail/messages/{unid}?format=mime

The mail service returns a response with the Content-Type header set to **message/rfc822** and a response body like this:

```
MIME-Version: 1.0
Subject: Address test -- outgoing mail from Raymond
To: DukeLawson@swg.usma.ibm.com,DeanMelnyk@swg.usma.ibm.com
Message-ID: <OF10C667B7.FFD62F61-ON852579A6.006864B1-852579A6.006876D9@LocalDomain>
From: RaymondChan@swg.usma.ibm.com
Date: Thu, 16 Feb 2012 14:01:01 -0500
Content-type: multipart/alternative;
    Boundary="0__=0ABBF335DFFBE2218f9e8a93df938690918c0ABBF335DFFBE221"
Content-Disposition: inline

--0__=0ABBF335DFFBE2218f9e8a93df938690918c0ABBF335DFFBE221
Content-type: text/plain; charset=US-ASCII

test
--0__=0ABBF335DFFBE2218f9e8a93df938690918c0ABBF335DFFBE221
Content-type: text/html; charset=US-ASCII
Content-Disposition: inline

<html><body>
<p><font size="2" face="sans-serif">test</font></body></html>
--0__=0ABBF335DFFBE2218f9e8a93df938690918c0ABBF335DFFBE221--
```

NOTE: The syntax for MIME messages is described in various standards documents including RFC2045, RFC2046 and RFC2047. Please consult the standards for more information. The JSON representation of a mail message is very similar to multipart MIME. Individual message parts have properties like contentType, contentDisposition, etc. It is beyond the scope of this document to describe this format in detail.

3.4 Sending a Message

To send a message, you send a POST request to the outbox resource URI. When sending a POST request, you must set the Content-Type header to **application/json** and include a request body like this:

```
POST /{database}/api/mail/outbox
Content-Type: application/json

{
  "to": [
    {
      "email":"DeanMelnyk@swg.usma.ibm.com"
    }
  ],
  "subject":"Sample plain text message",
  "content": [
    {
      "contentType":"text\/plain; charset=US-ASCII",
      "data":"This is a sample message body"
    }
  ]
}
```

The mail service parses the JSON input and sends the message to the address(es) listed in the **to** and **cc** properties. When the mail service successfully completes the request it returns HTTP status 201 (Created). Also the response includes a Location header referring to a copy of the message in the mail file's sent view. For example:

Location: http://server.xyz.com/mail/dlawson.nsf/api/mail/messages/5DCF8C13A0554131852579AC00593AB0

You can also send a message in MIME format. To do this, you set the request's Content-Type header to **message/rfc822** and include a request body like this:

```
MIME-Version: 1.0
To: DeanMelnyk@swg.usma.ibm.com
Subject: Sample plain text message
Content-Type: text/plain; charset=US-ASCII

This is a sample message body
```

Obviously, you can send more complex messages in either MIME or JSON format.

You can also send a reply to another message. To do this, you must specify the ID of message you are replying to. When sending a reply in JSON format, include the inReplyTo property like this:

```
"inReplyTo": "<OFC285B717.0EEFCF9B-ON85257C20.00542AB9-85257C20.00543D86@LocalDomain>"
```

When sending a reply in MIME format, include the In-Reply-To header field like this:

```
In-Reply-To: <OFC285B717.0EEFCF9B-ON85257C20.00542AB9-85257C20.00543D86@LocalDomain>
```

In either case, the ID must exactly match the messageId property (JSON) or Message-ID header field (MIME) of the "parent" message.

3.5 Reading the Sent and Drafts Views

To read the messages in the sent view, you follow the instructions in section 3.2 except you send a GET request to the sent resource:

GET /{database}/api/mail/sent

To read the messages in the drafts view, you send a GET request to the drafts resource:

GET /{database}/api/mail/drafts

The sent and drafts resources handle optional request parameters for paging, sorting, finding by key and searching. See section 3.2 for the list of request parameters.

3.6 Saving a Draft Message

To save a new draft message, you send a POST request to the drafts resource URI:

/{database}/api/mail/drafts

The details are similar to sending a message (described in section 3.4). You must include a Content-Type header with a value of **application/json** or **message/rfc822**, and you must include a request body in the corresponding format. The mail service parses the request body and attempts to save the draft message. When the mail service successfully completes the request it returns HTTP status 201 (Created). Also the response includes a Location header referring to the new message in the drafts view.

You can also update an existing draft message. To do this, you send a PUT request to the message URI – for example:

```
PUT /{database}/api/mail/messages/5DCF8C13A0554131852579AC00593AB0
Content-Type: application/json

{
  "to": [
    {
      "email":"DeanMelnyk@swg.usma.ibm.com"
    }
  ],
  "subject":"Sample plain text message",
  "content": [
    {
      "contentType":"text\/plain; charset=US-ASCII",
      "data":"This is a sample message body"
    }
  ]
}
```

Again, you must include a Content-Type header and you must include a request body in the corresponding format. If the mail service successfully updates the message, it returns HTTP status 200.

NOTE:

1. You can only update a message when it is currently in the drafts view. If you attempt to update a sent or received message, the mail service returns HTTP status 400 (Bad request).
2. When handling a PUT request, the mail service completely replaces the message. The current implementation has no support for partial updates.

3.7 Deleting a Message

To delete a message, you send a DELETE request to the message URI – for example:

DELETE /mail/dlawson.nsf/api/mail/messages/5DCF8C13A0554131852579AC00593AB0

If the mail service is able to delete the message, it returns HTTP status 200 (OK). By default the mail service performs a soft delete. In other words, it moves the message to the trash view.

You can also permanently delete a message by adding a parameter to the request. The following request permanently removes a message:

DELETE /mail/dlawson.nsf/api/mail/messages/5DCF8C13A0554131852579AC00593AB0?
permanently=true

3.8 Trash View Operations

3.8.1 Reading the Trash View

To read the messages in the trash, you send a GET request to the trash resource:

GET /{database}/api/mail/trash

The mail service responds with a list of messages in JSON format:

```
[
  {
    "href": "/mail/dlawson.nsf/api/mail/trash/8133974A3B9E27AF852574B8006B5A47",
    "from": {
      "displayName": "SAP Reporting"
    },
    "subject": "Reporting Lufthansa flights",
    "date": "2008-09-02T19:32:37Z"
  },
  {
    "href": "/mail/dlawson.nsf/api/mail/trash/06D44D8A0B708EE6852574B8007513EF",
    "from": {
      "displayName": "SAP Reporting"
    },
    "subject": "Reporting Lufthansa flights",
    "date": "2008-09-02T21:18:51Z"
  }
]
```

By default the mail service returns the first 10 messages in the trash. The response also includes a Content-Range header indicating the total number of messages. For example, the following header indicates the response contains messages 0 through 9 from a total of 21 messages.

```
Content-Range: items 0-9/21
```

To read more messages you can use the page and ps parameters as described in section 3.2.

3.8.2 Emptying the Trash View

To empty the trash you send a DELETE request to the trash resource:

DELETE /{database}/api/mail/trash

This permanently removes all of the messages in the trash.

NOTE: If there are several messages in the trash, this request may take a long time to execute. If there are hundreds of messages in the trash your HTTP client could time out before the request completes.

3.8.3 Permanently Deleting a Message in the Trash

To permanently delete a message in the trash you send a DELETE request to the trash message resource:

DELETE /{database}/api/mail/trash/{unique-id}

You should get the exact URI from the list of trash messages as described in section 3.8.1. For example, the following request permanently removes a message from the trash in /mail/dlawson.nsf:

```
DELETE /mail/dlawson/api/mail/trash/8133974A3B9E27AF852574B8006B5A47
```

4. Troubleshooting the Mail Service

The following sections list some common error conditions you might encounter. To help you troubleshoot the problem, each error condition is accompanied by one or more possible causes.

4.1 Bad Request

You send a request to a resource and the mail service responds with an HTTP status code of 400 (Bad Request). Usually this is caused by a bad URL parameter or, in the case of a POST or PUT, a poorly formatted request body. For example, consider this request:

GET /mail/dlawson.nsf/api/mail/inbox?ps=10&page=x

The mail service responds with a status code of 400 because page=x is invalid. Often the response body includes details about the error condition:

```
{
    "code":400,
    "text":"Bad Request",
    "message":"Invalid parameter page: x",
    "type":"text",
    "data":"java.lang.Exception: Invalid parameter page: x ...
}
```

4.2 Unauthorized

You send a request to a resource and the mail service returns an HTTP status code of 401 (Unauthorized). This usually occurs for one of two reasons:

1. You are sending an anonymous request (no user credentials) to a protected resource. In this case, you can correct the problem by supplying the appropriate user credentials with each request. The Domino server supports both basic authentication (HTTP Authorization header) and session authentication. See the appropriate Domino web server documentation for more about these authentication schemes.
2. The authenticated user has insufficient access to the resource. In this case, you can correct the problem by changing the access control list (ACL) for the database.

A response body in HTML format usually accompanies an HTTP status code of 401. The response body may help you troubleshoot the error:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Error</title></head>
<body text="#000000">
<h1>Error 401</h1>You are not authorized to perform this operation</body>
</html>
```

4.3 Forbidden

You send a request to a resource and the mail service returns an HTTP status code of 403 (Forbidden). This may be because the mail service is disabled for this server. See section 2 for instructions for enabling the mail service on a server.

4.4 Resource Not Found

You send a request to a resource and the mail service returns an HTTP status code of 404 (Not Found). This can be caused by an improperly formatted URI. For example, in the following request, the URI is incorrect:

```
GET /mail/dlawson.nsf/api/mail/draft
```

The URI is wrong because the word draft should be plural (drafts).

If you have checked the URI syntax and the Domino web engine is still returning an HTTP status code of 404, you should verify the mail and data service plug-ins are active in the OSGI runtime of the Domino server. From the Domino server console, type the following command

```
> tell http osgi ss
```

The response will be a list of OSGI plug-ins and their status. Verify that the following plug-ins are included:

```
com.ibm.domino.das_9.0.yyyymmdd-hhmm.jar
com.ibm.domino.services_9.0.yyyymmdd-hhmm.jar
com.ibm.domino.services.mail_9.0.yyyymmdd-hhmm.jar
com.ibm.wink_9.0.yyyymmdd-hhmm.jar
```

If these plug-ins are not included, the mail service has not been installed correctly on the Domino server. Please see the extension library installation procedure.

2.8.5 Could Not Connect

You send a request to a resource and the message "Error: Could not connect to server" is returned. Confirm the HTTP task is running on the Domino server. From the Domino server console enter the command:

```
> show tasks
```

Confirm the following line appears:

```
HTTP Server      Listen for connect requests on TCP Port:80
```

Please see the Domino Web Server documentation for additional information about troubleshooting HTTP connection problems.

2.8.6 Internal Server Error

You send a request to a resource and the mail service returns an HTTP status code of 500 (Internal Server Error). This should happen very rarely, if at all. As described in the previous sections, the mail service is designed to respond with status codes that help you pinpoint the cause of the problem. An HTTP status code of 500 could be caused by a software defect. Please report these errors to IBM.

Appendix A. Technical Notes

A.1 Relative vs. Absolute URLs

A previous version of the mail service returned responses with absolute URLs. For example, here is an excerpt from such a response:

```
    {
    
"href":"http:\/\/server.xyz.com\/mail\/dlawson.nsf\/api\/mail\/messages\/10C667B7FFD62F61852579A60068
64B1",
      "from": {
        "displayName":"Raymond Chan"
      },
      "subject":"Address test -- outgoing mail from Raymond",
      "date":"2012-02-16T19:01:01Z"
    }
```

Notice the value of the href property. It includes both the protocol and host address in the URL.

By default, more recent versions of the mail service return responses with relative URLs:

```
    {
      "href":"\/mail\/dlawson.nsf\/api\/mail\/messages\/10C667B7FFD62F61852579A6006864B1",
      "from": {
        "displayName":"Raymond Chan"
      },
      "subject":"Address test -- outgoing mail from Raymond",
      "date":"2012-02-16T19:01:01Z"
    }
```

In other words, the value of each href property is relative to the host address. This syntax is more compact and works better in certain configurations where mail service requests are proxied by other servers.

As an administrator, you can optionally configure the mail service to use absolute URLs (*not recommended*). Just add the following setting to the server's notes.ini file:

```
MailServiceAbsoluteUrls=1
```

.