MLINKADM-19.2

M-Link Administration Guide

Table of Contents

1 Software version

This guide is published in support of M-Link R19.2. It may also be pertinent to later releases. Please consult the release notes for further details.

2 Readership

This guide is intended for administrators who plan to configure and manage XMPP services using M-Link R19.2.

3 Typographical conventions

The text of this manual uses different typefaces to identify different types of objects, such as file names and input to the system. The typeface conventions are shown in the table below.

Object	Example
File and directory names	*/var/isode/log*
Program and macro names	isode.xmppd
Input to the system	`cd newdir`
Cross references	see Section 4, “Support queries and bug reporting”
Additional information to note, or a warning that the system could be damaged by certain actions.	Notes are additional information; cautions are warnings.

4 Support queries and bug reporting

A number of email addresses are available for contacting Isode. Please use the address relevant to the content of your message.

- For all account-related inquiries and issues: customer-service@isode.com. If customers are unsure of which list to use then they should send to this list. The list is monitored daily, and all messages will be responded to.
- For all licensing related issues: license@isode.com.
- For all technical inquiries and problem reports, including documentation issues from customers with support contracts: support@isode.com. Customers should include relevant contact details in initial calls to speed processing. Messages which are continuations of an existing call should include the call ID in the subject line. Customers without support contracts should not use this address.
- For all sales inquiries and similar communication: sales@isode.com.

Bug reports on software releases are welcomed. These may be sent by any means, but electronic mail to the support address listed above is preferred. Please send proposed

fixes with the reports if possible. Any reports will be acknowledged, but further action is not guaranteed. Any changes resulting from bug reports may be included in future releases.

Isode sends release announcements and other information to the Isode News email list, which can be subscribed to from the address: http://www.isode.com/company/subscribe.html

5 Export Controls

M-Link uses TLS (Transport Layer Security) to encrypt data in transit. This means that M-Link is subject to UK Export Controls.

For some countries (at the time of shipping this release, these comprise all EU countries, United States of America, Canada, Australia, New Zealand, Switzerland, Norway, Japan), these Export Controls can be handled by administrative process as part of evaluation or purchase.

For other countries, a special Export License is required. This can be applied for only in context of a purchase order from Isode.

The TLS features of M-Link are enabled by a TLS Product Activation feature. This feature may be turned off, and without this TLS feature M-Link is not export-controlled. This can be helpful to support evaluation in countries that need a special export license.

XMPP servers are generally deployed with TLS for security reasons and Isode strongly recommends that all operational deployments use the export-controlled TLS feature.

You must ensure that you comply with these Export Controls where applicable, i.e. if you are licensing or re-selling M-Link. M-Link is subject to an Isode license agreement and your attention is also called to the export terms of the license agreement.

Chapter 1 M-Link: Getting Started

This chapter discusses M-Link Server creation.

This chapter contains sections on the following topics:

1.1 Starting and Stopping M-Link Server

Once installed, the M-Link Server will start automatically as a system service, and thereafter the service can be controlled with the system-provided mechanisms.

On first start (and subsequently), M-Link will start with its configuration interface on port `5221`, ready to be activated and configured - this is accessible at `https://localhost:5221` on the server, or at `https://hostaddress:5221` from other machines. It will initially generate a self-signed certificate - both the certificate and the default port can be changed later through the configuration interface. The configuration interface can be accessed from a web browser running on a different machine from M-Link by substituting `localhost` with the hostname or IP of the server machine. As the initial certificate is self-signed, it will need to be manually trusted before a browser will accept it - usually by clicking through a security warning.

1.2 Creating an Initial Administrative User

M-Link Server can be configured by administrators. Administrators are users distinct from the users of the XMPP service. An initial administrator account is created as the first step through the web configuration interface - this user is stored in M-Link's configuration (the password is stored with a strong one-way hashing function, and cannot be recovered). Currently M-Link Edge Server is limited to a single administrator account.

1.3 Installing an Isode Activation Key

In order to use an M-Link Server instance on a host system, an activation key file is required. This file needs to be installed through M-Link's web browser-based configuration interface. The initial configuration process will lead you through generation of an Activation Request and the subsequent installation of the Activation Key supplied by Isode in response to receipt of the Activation Request.

Questions regarding licensing should be directed to support@isode.com.

1.4 M-Link Server Runtime User

When running on Unix, M-Link Server runs as an unpriviledged user, created during package install, and does not need to be started as 'root'.

On Windows, the M-Link Server runs as Windows service(s) under the `LocalSystem` account.

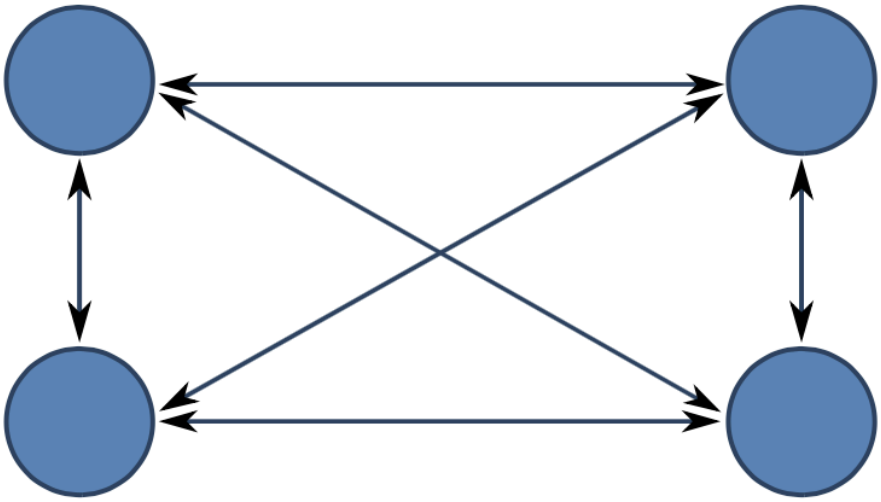
1.5 Adding Initial Configuration

Once you're logged into M-Link's Web Console and M-Link is activated you're ready to add initial configuration. Depending on the type of M-Link server that is being configured (and has been activated) you will need to either add domains for M-Link to host or add Peer Controls to allow routing between remote servers. Details of the configuration of XMPP Trunking and Peer Routing can be found in the Chapter 2, *Peer Controls and Links* chapter.

Chapter 2 Peer Controls and Links

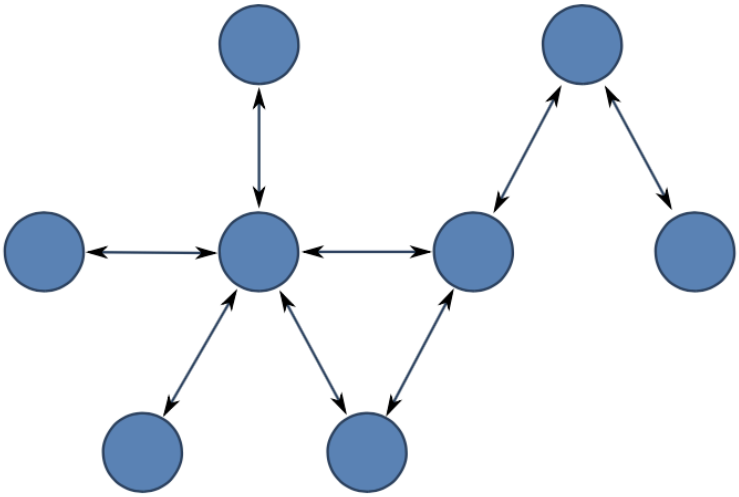
This chapter introduces the M-Link Edge Server product, explains what XMPP trunking is, and how M-Link Peer Controls can be used to support it. It also shows the use of *link*s to support operation with XML Guards.

Figure 2.1. Standard XMPP Server Configuration



In a standard XMPP configuration, XMPP servers are fully interconnected, as shown in Figure 2.1, “Standard XMPP Server Configuration”. This model works well for open XMPP deployments on the Internet. However, it does not work so well for cross domain, organizational boundary and constrained link scenarios.

Figure 2.2. XMPP Trunking



The model of XMPP Trunking, shown in Figure 2.2, “XMPP Trunking” extends the standard XMPP model to allow for configurations where servers are not fully connected. This structure enables XMPP messages to be switched through intermediate servers.

By default, when an XMPP server initiates a connection to another XMPP server (e.g., to other.organisation.example.com) it will look up the domain in the *Domain Name*

System (DNS), first with SRV records and falling back to A/AAAA records, and connect using the IP address determined from this lookup. This leads to the fully connected approach.

Peer Controls are the mechanism used by M-Link to enable configuration of an XMPP Trunking architecture. They are checked prior to the standard DNS lookup, and provide M-Link with information on how to connect to a specific domain. Each peering control relates to a specific remote domain (e.g. other.organisation.example.com), to a collection of subdomains (e.g. all subdomains of "isode.com"), or a default catch-all. A peering control can direct the connection to a list of hosts and/or IP address or specialised connection mechanism configured as part of the peering control. Peer controls can be used to reject the traffic to another site entirely, require particular security and authentication, and in some M-Link products restrict it to certain types and control the network connections used for the traffic.

Further information on XMPP Trunking and use of M-Link Peer Controls is provided in the Isode white paper Providing XMPP Trunking with M-Link Peer Controls [https://www.isode.com/whitepapers/xmpp-trunking.html].

2.1 M-Link Edge Server

M-Link Edge Server is an M-Link server with no local users that is used in boundary and cross-domain configurations to check and potentially modify traffic. M-Link Edge Server will often be used in conjunction with an XML Guard to provide cross-domain protection and may use *XEP-0361* or *GCXP* to communicate with the XML Guard.

M-Link and M-Link Edge Server are both provided by the same underlying technical product, but are sold as different products. Note that *XEP-0361* may not be used with a standard M-Link deployment that supports local users.

2.2 Peer Controls

Peer Controls define the rules for communicating with remote domains, and are checked prior to DNS being used. A Peer Control can be defined using one of three Matching Rules:

Domain
: A single specific domain. This is used to force routing for a specific domain.

Subdomains
: All subdomains of the domain, but not the domain itself. If this Matching Rule is selected for a Peer Control for `example.com`, domains that will match include `pubsub.example.com` and `chat.private.example.com`.

Domains and Subdomains
: The domain itself, and all its subdomains. If this Matching Rule is selected for a Peer Control for `example.com`, domains that will match include `example.com` and `pubsub.example.com`.

When multiple Peer Controls match a certain remote domain, the longest match is the one that will take effect. When no match is found for the domain, the *default* Peer Control will be applied.

This combination provides full flexibility to configure arbitrary XMPP trunking configurations.

When a peer control is in place, it is possible to configure specific actions for the peer:

- Chapter 4, *Security Labels, Clearances and Policies* checking against a configured security clearance for the peer.
- Section 2.3, “Peer Authentication” options.
- Section 2.4, “Relay zones”, which allows for relaying stanzas between servers.
- Traffic filtering using Section 2.5, “Transformations”, which enables removal or modification of traffic to and from the peer.

Traffic to a remote domain will by default use S2S connections which are negotiated between the servers. If the option *Use Specialised Connection Mechanism* is enabled, communication will be sent and received using the specified link (see Section 2.6, “Links”).

SRV lookups for S2S can be overridden by enabling the *Override DNS* option and specifying values for the *Connect Host* and *Connect Port* options. *Connect Host* can either be an IP address, or a hostname. Multiple *Connect Host* and *Connect Port* pairs may be specified, with connection attempts being made in the order in which the pairs are specified.

If the option *Use Specialised Connection Mechanism* is enabled, communication will be sent and received using the specified link.

2.3 Peer Authentication

Authentication of remote servers is usually based on the addressing of the XMPP data that are being sent, not on the DNS name of the server hosting the remote service (although these are sometimes the same). There are two mechanisms for authenticating server to server (S2S) connections: Dialback (*XEP-0220*) and TLS authentication (SASL EXTERNAL).

2.3.1 TLS Authentication

TLS Authentication is based on the certificate presented by a remote server during TLS negotiation, and could take four forms.

- Normal. Absent further configuration, TLS Authentication is performed by checking the subjects of the certificate provided by the remote server, and if they match the server address (or Remote Host if a Link is in use) of the XMPP data that are being transmitted, checking to see if the certificate was issued by a trusted root certificate. For a detailed discussion, see Section 3.2, “Certificate Verification” and Section 3.3, “Trust Anchors”
- Connect Host. If `Peer Connections` is enabled through "Override DNS" and normal TLS Authentication is attempted and fails, M-Link will perform the same authentication checks:
  - Outbound. Using the hostname from the `Peer Connections` list which has resulted in the successful connection.
  - Inbound. Using each hostname from the `Peer Connections` list in turn.
- Remote Host. If and X2X or GCXP link is being used, M-Link will use the Remote Host value of the Link as the subject to check.
- Pinned Certificates. If a pinned peer or link certificate is set then TLS Authentication is performed only by comparing the certificate provided by the remote server to the one in this option.

2.3.2 Dialback

Dialback uses checks against the entries in DNS to authenticate a remote server (connections that use dialback for authentication may also use unauthenticated TLS,

and if `TLS` is set to `Required`, must). This is generally considered a weaker form of authentication than TLS provides. Authentication must happen in both directions (that is: we must authenticate that the remote server is who they claim to be, and the remote server must authenticate that we are who we claim to be), and dialback is controlled independently for each direction. If `TLS` is set to any value other than `Require Authenticated TLS` we will be willing to authenticate a remote server's identity using dialback. If the `Perform Dialback When Offered` option is enabled then we will allow a remote server to authenticate our identity using dialback.

Where there is a choice of TLS Authentication or Dialback, M-Link will always prefer to use TLS Authentication.

2.4 Relay zones

Relay zones are the M-Link mechanism for allowing relaying of stanzas between servers in different logical groups (generally by bridging a network boundary of some sort, such that the servers would not be able to establish a connection otherwise). When receiving a stanza from a peer if the 'to' domain is not serviced by the local server a peer control matching the 'to' will be searched for and, if found and if the peer control of the 'to' gives a different relay zone from the peer control of the connection from which the stanza is received, it will be routed out to the target peer. This means that M-Link will not relay stanzas between servers within the same relay zone (because they should be communicating directly, or simply shouldn't be communicating).

2.5 Transformations

This section describes transformations that may be configured for a Peer Control.

Transformations can be applied to stanzas coming in from or going out to peers. Each Peer Control can have a number of these Transformations, which are executed in the order they are listed. One typical case where this may be useful is when network capacity is constrained: for example, on a slow link it may make sense to set up a transformation to strip all Chat State Notifications.

Transformations are selected from the Transformations object store. Each transformation is an *XSLT 1.0* document operating on a single stanza in the `jabber:server` namespace, wrapped in a minimal XMPP Server-to-Server stream element. The result of a transformation should also be a single stanza wrapped in a stream element.

M-Link ships with transformations for common operations, such as stripping XHTML-IM (*XEP-0071*) elements from messages, or normalizing XML for communications to M-Guard. These transformations are always present, and cannot be removed or renamed, however uploading variants of the built-in rules with modified behaviour is possible. A complete list of all transformations shipped with M-Link can be found in Appendix A, *Pre-defined Transformations*.

The wizard which allows user-defined *XSLT 1.0* documents to be uploaded to the Transformation object store also allows simple transformations which strip namespace / element combinations to be created and uploaded.

Example 2.1. A stanza wrapped for transformations

```
<stream:stream id='1' version='1.0' xmlns='jabber:server'
    xmlns:stream='http://etherx.jabber.org/streams'>
<message from='anne@example.com' to='bob@example.net' type='chat'>
```

```
  <body>Hello!</body>
  <active xmlns='http://jabber.org/protocol/chatstates'/>
</message>
</stream:stream>
```

Example 2.2. An XSLT that will strip Chat State Notifications from message stanzas

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:stream='http://etherx.jabber.org/streams'
    xmlns:xmpp='jabber:server'
    xmlns:xhtml-im='http://jabber.org/protocol/xhtml-im'
    xmlns:csn='http://jabber.org/protocol/chatstates'>

 <xsl:output method="xml" omit-xml-declaration="yes"/>

 <xsl:template match="@*|node()">
  <xsl:copy><xsl:apply-templates select="@*|node()"/>
 </xsl:copy></xsl:template>

 <xsl:template match="/stream:stream/xmpp:message/csn:active"/>
 <xsl:template match="/stream:stream/xmpp:message/csn:composing"/>
 <xsl:template match="/stream:stream/xmpp:message/csn:paused"/>
 <xsl:template match="/stream:stream/xmpp:message/csn:inactive"/>
 <xsl:template match="/stream:stream/xmpp:message/csn:gone"/>
</xsl:stylesheet>
```

If the original stanza had children, but the resulting stanza no longer has any, the option *Allow Childless Message and Presence Stanzas Through* determines whether the stanza is let through. For example, if the stanza in example Example 2.3, “A stanza with only Chat State Notifications contents” is sent to a peer that has a Transformation rule that removes CSN (as shown in Example 2.2, “An XSLT that will strip Chat State Notifications from message stanzas” and the option *Allow Childless Message and Presence Stanzas Through* set to *Don't let stanzas through* or *Only allow presence stanzas through*, this stanza will not be sent to the recipient.

Example 2.3. A stanza with only Chat State Notifications contents

```
<stream:stream id='1' version='1.0' xmlns='jabber:server'
    xmlns:stream='http://etherx.jabber.org/streams'>
<message from='anne@example.com' to='bob@example.net' type='chat'>
  <composing xmlns='http://jabber.org/protocol/chatstates'/>
</message>
</stream:stream>
```

If the transformation returns an empty or invalid document, or a document that does not parse into a syntactically valid stanza, or the test described in the previous paragraph blocks the stanza, what happens is determined by the *Action on Blocked Stanza* option associated with the transformation. *Bounce* will send back an `error` stanza to the source of the stanza if appropriate for the stanza. *Drop* will cause the stanza to be discarded without notifications.

2.6 Links

A Link defines a specialised connection mechanism for federation. Configuration of connection mechanisms and authentication/security live within the Link configuration for all Peers that use links, rather than in the Peer Control. M-Link currently supports these types of links:

X2X
: *Zero Handshake Server to Server Protocol (XEP-0361)* operating over TCP. This protocol is designed to support constrained bandwidth links with high latency, where standard S2S will lead to performance problems.

GCXP
: *Guard Content Exchange Protocol*, operating over TCP. This protocol is used for communications between M-Link and M-Guard.

  By default, GCXP links are configured to send and expect to receive GCXP responses, and M-Guard should be configured to let GCXP responses through. In setups where M-Guard cannot be configured to let GCXP responses through, the option *Send GCXP Responses* must be disabled as well. GCXP responses are used to provide acknowledgement of stanzas delivered between the M-Link Edge Server servers on either side of the Guard, allowing stanza delivery to be re-attempted in the case of a connection or system failure of the Guard or other M-Link Edge Server. They are also used by M-Guard to alert M-Link Edge Server when a stanza has been rejected by the Guard, allowing M-Link Edge Server to 'bounce' the stanza. GCXP responses contain no data other than the ID of the GCXP message to which they refer; connections to the Guard remain unidirectional with respect to the flow of XMPP data.

Note: Peers using links that communicate using this protocol will generally require the *Normalize XML for M-Guard* Transformation.

2.6.1 Connection parameters

All links can take the following parameters:

Local Port
: TCP port the link will listen on for new connections. The server will try to bind to this port on all local IP addresses.

Fallback Link
: This option points to the next link that should be tried when an attempt to establish a connection fails. When this parameter is unset, no further links will be tried. This can be used to create a list of links that will be tried in order when connecting to a peer.

Note: Peer Controls using a link with fallbacks must always point to the first link in the chain.

Listen Only
: When this option is set, the server will never initiate a connection over this link on its own, but will accept connections coming in. Established *Listen Only* links are always *Bidirectional*.

Bidirectional Connections
: If set, a single connection can send XML stanzas in both directions. If not set, a connection will be established for each direction of stanza flow. This option is only available on *XEP-0361* links, GCXP links are never bidirectional.

Remote Host
: For non-Listen Only links, this indicates which host to connect to. This might either be an *Fully Qualified Domain Name* (FQDN), or an *IP address*.

Remote Port
: For non-Listen Only links, this indicates which port on Remote Host to connect to.

Enable XEP-0198
: When this option is enabled, the link will use Stream Management (*XEP-0198*) to ensure stanzas sent are received by the remote server. This will also enable acknowledgements of stanzas received from the remote server.

XEP-0198 Version
: This option selects which version of *XEP-0198* will be enabled on this link. By default, this will use the most recent version of the standard as supported by M-Link. When interoperating over a Stream Management enabled link with a server running a version of M-Link older than R19.0 , this option should be set to `Legacy XEP-0198`.

Use TLS
: When this option is enabled, and if a certificate is configured, the link will use and require TLS.

When Use TLS is enabled, the following additional options are available:

TLS Identity
: Key and certificate this link presents to inbound connections, as well as sends to the peer when trying to establish a connection.

Pinned Link TLS Certificate
: If set, this link will only allow connections with peers that present this exact certificate.

Require a Valid Certificate in TLS Session
: When set, peers must use a valid certificate that isn't expired, matches the Remote Host, and when Pinned Link TLS Certificate is set, matches the pinned certificate.

  When this option is not set, any TLS certificate will be accepted, and TLS will only be used to prevent eavesdropping.

Ignore System Trust Anchors
: When set, only trust anchors explicitly configured within M-Link are trusted, and system trust anchors will be ignored.

Link Trust Anchors
: This is a list of TLS Trust Anchors that will be used for Section 2.6.2, “Authentication”.

2.6.2 Authentication

When using Links, authentication is based on the identity of the server to which the Link connects, not to the addressing of the XMPP data that are being transmitted. Unless the requireValidCertificates option is enabled on a Link (and TLS is configured) any connection associated with the Link is considered to be authenticated for transmission of XMPP data addressed to any Peer Control configured to use the Link. If the requireValidCertificates option is enabled then the certificates used in TLS must be trusted - either by the pinnedLinkCertificatePEM option matching the provided certificate, or by the presentation of a certificate whose subject (see Section 3.2, “Certificate Verification”) matches the remoteHost option and was issued by an authority configured as a trust anchor (see Section 3.3, “Trust Anchors”).

Chapter 3 TLS Configuration

This chapter describes how M-Link is configured to use TLS, covering configuration of certificates/keys used by the local server, and trust of certificates used by other servers.

Note: Note: Your M-Link installation must have the TLS-HGE feature enabled before TLS can be used.

Transport Layer Security (TLS) is used by M-Link to provide privacy, data integrity and authentication for client-to-server and server-to-server connections. Use of TLS requires the configuration of three main classes of information:

Public Key Certificate Chain
: A Public Key Certificate Chain certifies the ownership of a public key (by the named subject of the certificate).

Private Key
: The Private Key corresponding to the Public Key Certificate Chain noted above.

Trust Anchors
: These are a set of certificates which are to be treated as authoritative entities for which trust is assumed.

A Public Key Certificate Chain and corresponding Private Key must be configured before M-Link can act as a TLS server (that is to say, the responding side of a TLS handshake) and provide a TLS-encrypted connection.

The same Public Key Certificate Chain and Private Key will be used by M-Link when it is acting as a TLS client (initiating a TLS exchange with a TLS server).

Whether the Public Key Certificate Chain provided by M-Link in the TLS handshake also provides authentication depends on whether the Certificate Chain has been signed by a Trust Anchor which the other party to the TLS handshake trusts.

The Trust Anchor list identifies a set of trusted signing entities which will be used when verifying the validity of a Certificate Chain. The configured list may be used in addition to Trust Anchors provided externally to M-Link (installed as part of the operating system, or placed in system-wide certificate stores), or can replace these external Trust Anchors completely.

Private Keys are normally stored in encrypted form. In this case, a passphrase which will be used when decrypting the Private Key must also be configured.

A number of secondary TLS controls are available:

Disable TLSv1
: This allows use of TLS protocol version 1 to be disabled. This defaults to false, so that TLSv1 is allowed.

Cipher Suites
: The set of cipher suites which will be proposed or accepted during TLS handshaking can be configured, as a standard OpenSSL cipher string. This is empty by default, allowing any supported cipher suites to be used.

Diffie-Hellman Parameters
: A set of Diffie-Hellman parameters must be provided before some classes of cipher suite can be used by M-Link. A default set of parameters is built in, but can be overridden if necessary.

Ignore System Trust Anchors
: Disables use of built-in Trust Anchors; only those Trust Anchors which have been configured for M-Link will be used. This defaults to false.

Public Key Certificate Chains, Private Keys and Trust Anchors are configured as PEM-encoded strings. Many Certification Authorities provide certificates in this form anyway, and tools are readily available to convert certificates which have been provided in other formats.

3.1 Detailed Configuration

TLS information can be specified at various points within M-Link's configuration, as described below.

3.1.1 Default Settings

Default TLS configuration can be specified for an M-Link server as a whole. In a simple setup this may be all that is required:

Public Key Certificate Chain

Private Key

Private Key Passphrase

Trust Anchors

Disable TLSv1

Cipher Suites

Diffie-Hellman Parameters

Ignore System Trust Anchors

3.1.2 Peer Control Settings

Peer control TLS settings have the effect of overriding the TLS settings for the IM or MUC domain domains to which they apply (and the default settings), when performing server-to-server (S2S) communication:

Public Key Certificate Chain

Private Key

Private Key Passphrase

Trust Anchors

Ignore System Trust Anchors

A Peer Control may also specify the following TLS-specific settings:

TLS
: This item controls how TLS is used by the S2S link, and takes the values:

- Never - do not use TLS
- Available - use TLS encryption if available
- Required - require use of TLS encryption
- Required With Auth - require an authenticated TLS connection

Pinned Peer Certificate
: See below for a detailed description

A Pinned Peer Certificate allows the standard certificate verification (performed during the TLS handshake) to be overridden. Instead of the certificate presented by the Peer being verified against Trust Anchors, checked for revocation etc, a simple comparison with the Pinned Certificate is performed, and if the presented and Pinned certificates match, the authentication is successful.

3.1.3 Link Settings

Link TLS settings have the effect of overriding any default TLS settings:

Public Key Certificate Chain

Private Key

Private Key Passphrase

Trust Anchors

Ignore System Trust Anchors

In addition, a Link may specify:

Use TLS
: If this is set true, TLS will be used and required for this Link.

Pinned Link Certificate
: See below for a detailed description

Require Valid Certificate
: If this is set true, a TLS session must present a valid certificate.

A Pinned Link Certificate allows the standard certificate verification (performed during the TLS handshake) to be overridden. Instead of the certificate presented by the Link being verified against Trust Anchors, checked for revocation etc, a simple comparison with the Pinned Certificate is performed, and if the presented and Pinned certificates match, the authentication is successful.

3.2 Certificate Verification

A certificate which is presented as part of a TLS handshake is verified via a multi-stage process. The first stage takes place during the handshake itself, and checks (among other things) that the certificate has been signed by one of the Trust Anchors which have been configured for the current TLS context. Even if the certificate fails one or more of these checks, the TLS handshake may still complete successfully, with an encrypted TLS session being established.

Once the TLS handshake is complete, secondary checks are performed on the presented certificate, if the configuration of the domain or link requires a valid certificate. These are:

- If a Pinned Certificate is configured, the results of the first-stage verification are ignored and a direct comparison between this and the presented certificate is performed. If they match, the presented certificate is considered valid. If they do not match, the presented certificate is considered invalid. In either case, no further verification of the presented certificate is performed.
- A check of the result of the first-stage verification, described above. If this first-stage verification has failed, no further action is taken, and the certificate is considered invalid.
- For Server-to-Server connections which use a Peer Control, a Subject Alternative Name match is attempted for the XMPP domain to which the connection is being made. If this fails, a Subject Alternative Name match for the Peer Control's connectHost configuration setting is attempted. If both of these matches fail, the certificate is considered invalid.
- For Server-to-Server connections which use a Link, a Subject Alternative Name match against the Link's remoteHost configuration setting is attempted. If this match fails, the certificate is considered invalid.

- If the appropriate checks described above succeed, the certificate is considered valid.

A certificate can contain multiple Subject Alternative Names, of varying types. When attempting to match a domain name, DNS name or hostname against these, a number of different comparisons are performed:

- A match against one of the certificate's DNS Names. This includes wildcard matching, so that a certificate with a DNS Name of `*.isode.com` would match `mary.isode.com`.
- A match against one of the certificate's SRV Names. SRV Names are prefixed with `_xmpp-server` if the certificate belongs to an XMPP server or `_xmpp-client` if presented by an XMPP client. Thus an SRV name of `_xmpp-server.mary.isode.com` would match the domain `mary.isode.com`.
- A match against one of the certificate's XMPP Addresses.
- If the certificate has no other Subject Alternative Names, a match against one of the certificate's Common Name values.

3.3 Trust Anchors

Trust Anchors are certificates which identify trusted signing entities. These are used by M-Link to verify that a chain of certificates (up to and including an end-entity certificate) received from another XMPP server or client is valid.

Most operating systems provide a built-in set of Trust Anchors which identify commercial Certification Authorities. The location and format of these is system-specific. By default, M-Link will make use of these Trust Anchors. Use of system Trust Anchors can be overridden via configuration either for the whole M-Link installation or per-Peer Control or per-Link.

A set of private Trust Anchors may be specified as part of M-Link's configuration, either across the whole installation, per-Peer Control or per-Link. Use of private Trust Anchors is required when the end-entity certificates being presented have been signed by Certification Authorities whose CA certificates are not configured as part of the operating system.

Chapter 4 Security Labels, Clearances and Policies

This chapter describes how Security Labels, Clearances and Policies are configured and used within M-Link.

A Security Label is a structured representation of the sensitivity of a piece of information (e.g. a message). This is used in conjunction with a Security Clearance (a structured representation of what information sensitivities an entity is authorized to access) and a Security Policy to control access to each piece of information. For instance, a message stanza could be labelled as "SECRET", hence requiring the sender and the receiver to each have a clearance granting access to "SECRET" information. Labels include a Display Marking which provides a human-readable form of the machine-readable label, as well as information about which colours XMPP clients should use when displaying the label.

The use of Security Labels in XMPP is described in detail in XEP-0258: Security Labels in XMPP. General information on Isode's Security Policy Infrastructure can be found at Isode Security Policy Infrastructure and its use by Isode within XMPP is described in Using Security Labels to Control Message Flow in XMPP Services.

Determining whether a labelled message stanza is permitted to be transferred to an entity (i.e. a domain or person) is a two stage process:

- The label associated with the destination entity is tested against the clearance associated with the message stanza's sender entity. This is testing "Is the sender of this message cleared to send a message to this entity?".
- The label on the message stanza is tested against the clearance associated with the destination entity. This is testing "Is the recipient of this message cleared to receive a message with this security label?".

Each of the tests described above takes place within the context of a Security Policy. The Security Policy controls how the Label/Clearance comparison is performed, and may provide other facilities such as equivalences to other Security Policies. When Security Label conversion is taking place, two Security Policies will be used: the Policy under which the Label was issued and the Policy to which the label will be converted.

Each entity which originates or accepts message stanzas may be configured with its own Security Label and Clearance. A system of defaulting within the M-Link configuration means that, in the simplest case, a single server-wide Security Label and Clearance are all that need to be configured. For more complex systems, individual entities can override defaults with their own settings.

A final configuration item which may be associated with each entity is a Default Stanza Label. This is the Security Label which will be applied to any unlabelled message stanza originating at that entity.

Security Labels and Clearances are maintained within catalogs (a Label Catalog is a collection of associated labels; a Clearance Catalog is a collection of associated clearances). Multiple label and clearance catalogs may be loaded into an M-Link configuration. Specific label and clearance catalogs may then be associated with individual components of the server's configuration, and individual Security Labels and Clearances selected for use from within these catalogs.

4.1 Object Stores

Security Policies, Security Label catalogs and Clearance catalogs are maintained in type-specific Object Stores within M-Link. All three types of entity are usually maintained and distributed publicly as XML documents. The M-Link Web Console (MLWC) allows these XML documents to be loaded into the appropriate Object Stores. Catalogs, Labels and Clearances within catalogs, and Security Policies can then be selected for use in configuration options.

Security Policies are represented within M-Link as SDN.801c Security Policy Information Files in Open XML SPIF format. Isode provides sample Security Policies and associated Security Label and Clearance catalogs in subdirectories of $(SHAREDIR)/security_label/example_data.

4.2 Server Configuration

The global "Use Security Labelling" option within MLWC controls whether any Security Labelling configuration is exposed and enabled.

> **Note:** Note: If Security Labelling has been enabled, but no Clearance has been configured for the Server, all messages which contain Security Labels will be blocked.

Within the Security Labelling Configuration section of MLWC, the following items may be configured:

Server Default Security Policy
: The default Security Policy to be used by the server. This is a selection from the set of policies which have been loaded into the Security Policy Object Store.

Server Default Security Label
: This is the default Security Label to be used by the server. It is selected in a two-stage process; first a Security Label Catalog is chosen from the Security Label Catalog Object Store, and then an individual Security Label is chosen from within that catalog. This allows the transit of messages from originators with unsuitable clearances to be blocked.

Stanza Default Security Label
: This is the default Security Label which is applied to unlabelled stanzas. It is selected in a two-stage process; first a Security Label Catalog is chosen from the Security Label Catalog Object Store, and then an individual Security Label is chosen from within that catalog. The Security Label Catalog defaults to that chosen for the Server Default Security Label.

Server Default Security Clearance
: This is the default Security Clearance to be used by the server. It is selected in a two-stage process; first a Security Clearance Catalog is chosen from the Security Clearance Catalog Object Store, and then an individual Security Clearance is chosen from within that catalog. The Clearance blocks the transit of messages without a suitable label.

Default XEP-0258 Label Format
: This controls the encoding in which Security Labels will be emitted.

Require Security Label on Inbound Message
: This mandates that a Security Label is present on all inbound message stanzas.

Manually Configured XEP-0258 Catalog
: This contains a label catalog in XEP-0258 format which will be served to clients, telling them which labels they can apply to messages which they send. If this item is not configured, a label catalog in XEP-0258 format will be automatically generated from the Security Label Catalog configured for the client's domain.

4.3 Peer Control Configuration

Individual Peer Controls may have their own Security Labelling configuration. This provides a mechanism for controlling the classification of messages which can flow to and from an external domain. For example, a simple configuration might allow messages labelled as Unclassified, Sensitive or Confidential to be sent between users within the same local IM domain, but only permit Unclassified messages to be passed to external domains.

Individual Peer Controls may also have their own Security Policy configured. When this is the case, messages flowing to the Peer Domain will have their labels converted so that they are consistent with Security Policy which has been configured for the Peer.

If an inbound message includes a label which has not been issued under either the Server Security Policy or the Peer's own Security Policy (if configured), then the label will be discarded and the message will be treated as if unlabelled. An Audit record will be generated to record the label discard.

The following items can be configured:

Peer Security Label
: This is the Security Label applied to the Peer: only stanzas with a source clearance which permits access to this Security Label will be transferred to the Peer. The label is selected in a two-stage process; first a Security Label Catalog is chosen from the Security Label Catalog Object Store, and then an individual Security Label is chosen from within that catalog. When setting this option, the catalog choice will default to that selected for the Server as a whole, but the Peer Security Label does not have a default and must be explictly selected, since it would normally be different from that of the Server.

Peer Default Stanza Security Label
: This is the default Security Label which is applied to unlabelled stanzas arriving from this Peer. It is selected in a two-stage process; first a Security Label Catalog is chosen from the Security Label Catalog Object Store, and then an individual Security Label is chosen from within that catalog. The Security Label Catalog choice defaults to that selected for the Server as a whole, and the Peer Default Stanza Security Label choice defaults to the Server's Stanza Default Security Label.

Peer Security Clearance
: This is the Security Clearance applied to the Peer: only stanzas which include a label which the Peer's clearance allows will be transferred to the Peer. The clearance is selected in a two-stage process; first a Security Clearance Catalog is chosen from the Security Clearance Catalog Object Store, and then an individual Security Clearance is chosen from within that catalog. The Clearance Catalog defaults to that selected for the Server as a whole, while the Peer Security Clearance itself must be explicitly selected, since it would normally be different from that of the Server.

Peer Outbound Default Stanza Security Label
: This allows the configuration of a Security Label which will be applied to all unlabelled outbound messages for this Peer.

Peer Relabelling Security Policy
: This is the Security Policy which applies to the Peer. If this is set (and different from the Server Default Security Policy), the label on an outbound message (issued

under the Server Default Security Policy) will be replaced with an equivalent label issued under the Peer Relabelling Security Policy. For this to be possible, the Server Default Security Policy must have equivalencies for the Peer Relabelling Security Policy configured.

Peer Outbound Security Label Format
: This controls the format in which outbound Security Labels are emitted. It defaults to the server-wide Default XEP-0258 Label Format setting.

Peer Update Display Markings
: Security Labels include a display marking phrase (e.g. "Secret") and optionally foreground and background color specifications. The default behavior of M-Link is to normalize these fields when serializing a label, replacing values present in the label with those defined for the label by the Security Policy. This control allows this functionality to be disabled, so that labels are emitted with the display markings that they arrived with.

Require Security Label on Inbound Messages
: This mandates that a Security Label is present on all inbound message stanzas from a Peer.

Peer Mandatory Label
: This allows configuration of an IC-ISM format label which will be required by default to be present on all inbound message stanzas from the Peer. Messages which do not contain a matching label will be rejected. A label match will cause the label to be removed from the message stanza and the Peer Default Stanza Security Label (if any) to be applied to it. The label may also be applied to outbound messages to the Peer. This is a specialized option and should only be used when suggested by Isode Support.

Require Peer Mandatory Label on Inbound Messages
: The default value of this option requires the label configured in the Peer Mandatory Label option to be present in all inbound messages.

Apply Peer Mandatory Label to Outbound Messages
: This causes the label configured in the Peer Mandatory Label option to be applied to all outbound messages, replacing any other label present.

Add FLOT to Outbound Messages
: This configures the conditions under which a First Line Of Text marking is added to the body of messages, and is intended for use when sending messages to Peers which do not support XEP-0258 labeling. The FLOT marking added is derived from the message's Security Label. If the message also includes an HTML payload, this will be stripped. The "Add FLOT if message has non-default Label" option will cause the Security Label on the message to be compared against the Peer Outbound Stanza Default Security Label, if set, otherwise against the label marked as Default in the Security Policy, if any. If a match is detected, no FLOT will be inserted.

Peer Fixed Security Label
: This configures a fixed Security Label (chosen from a Label Catalog) which will be applied to all outbound messages, replacing any Security Label present on the message.

Peer Disable Label Out
: If set to true, outbound messages will not include a security label. Access control checks will still be performed if configured.

4.4 Audit Logging

Access Control Decision Function (ACDF) failures are audited by M-Link. Two different audit types are generated:

ACDFSourceFailure
: This indicates that either the source of the stanza did not have a Clearance, or that the processing entity's Label is not permitted by the Clearance.

ACDFLabelFailure
: This indicates that the stanza's Label is not permitted by the processing entity's Clearance.

In both cases, the stanza's To and From address fields are included in the audit record, together with the name of the domain for which the ACDF is being called. In the case of an unexpected ACDF failure, this should enable the identification of configuration point responsible for the failure. A free-form error string indicating the reason for the ACDF failure is also included.

Chapter 5 M-Link Configuration and Management

This chapter introduces the M-Link Web Console, used for configuration and management of the M-Link server.

M-Link supports configuration and management through the M-Link Web Console, including management of product activation keys. M-Link has a built-in web server accessible to a browser by giving the server host as URL and specifying the “Web Admin Port”. By default, that is 5221, for example:

```
https://localhost:5221/
```

This example assumes that HTTPS is used for the Admin interface, which is the default unless the TLS feature is not available. Alternatively:

```
http://localhost:5221/
```

It is also possible to configure M-Link programatically using the interface defined in Chapter 7, *HTTP API*.

M-Link cannot be used without an activation key. Upon first use, the M-Link Web Console takes the user through product activation and initial administrator user setup.

- Product activation key management is addressed later in this chapter.
- Initial administrator user credentials are included within the M-Link Web Console configuration, to allow set up independently of any external directories. Their input is a one-off set up.

If the operations listed above have been performed, the M-Link Web Console starts with the login page.

Once successfully logged in, the “home page” of the M-Link Web Console exposes the “Global Options” and, on the left hand side, a menu of configuration and monitoring sections. These are briefly described in following sections.

5.1 Common Features

5.1.1 UI Navigation

The M-Link Web Console consists of a home or root page, displaying “Global Options”, and a number of sections or sub pages, accessible by clicking in a menu bar on the left hand side. These are organized in a tree-like structure, each page having a path from the root. The side bar menu adjusts to the new location when navigating to different pages, and provides means of stepping back on the path from the root. The “Back” browser action also works, but that returns to the previous page, not necessarily a page closer to the root. In addition, when viewing sub pages, a representation of the path to the page is displayed under the page title in the header bar. It includes clickable elements which also allow backwards navigation.

Every navigation device (button, card or link which, upon mouse click, takes the M-Link Web Console to a different page), also supports “Open Link in New Tab” and “Open Link in New Window” normally provided by browsers in their context menus.

When working with multiple servers, it may be useful to identify tabs/windows connected to the same server: see option “Application Title Override” at the bottom of the “Global Options”.

5.1.2 Viewing and Editing Settings

Each option is displayed with a name and followed by a description. Please refer to other chapters of this manual for the full significance of each option. Sometimes, long descriptions appear shortened. These can be viewed in full by clicking on the `More...` link.

Most options are defined by a simple value which can be edited in various ways as appropriate: mouse clicks, text input or drop-down menu option selection. Changes only take effect, that is, are conveyed to the server, when the Submit button, at the bottom of the page, is clicked. There is also a Cancel button which restores the page to the values currently set in the server.

Some options are only visible and in use when enabled by other settings.

- This serves to remove clutter, such as option “Customise Ports”, which needs to be selected before port number options appear.
- It also helps to manage dependencies: some option combinations would not always make sense. When options are thus controlled from another page, it may be necessary to submit the page with the enabling option before the dependent features appear. For example, the “Security Labelling Configuration” section only appears after submitting the “Global Options” page with “Use Security Labelling” ticked.

A few options are more complex and have a specialized editing mechanism. These are described later.

5.1.3 Default Values

A large number of options have defaults. These are denoted by a value displayed in italics, unless it is a boolean option (i.e. selected or not). They will also have a “Use default” tick box.

- This can be ticked to restore the option to its default value. If the tick is showing, then the option is definitely set to its default.
- Changing the value will automatically untick the box. It is, of course, possible to set values which happen to equal the defaults, in which case the server remembers this state explicitly, and the configured values will persist even if the defaults are changed.

Default values are often immutable, possibly defined by a standard. In some cases the default is itself an option which can be modified. Any option used as a default elsewhere is displayed with a message including paths to all the options currently defaulting to that option.

5.1.4 Validation

Option values are generally constrained by their types. For example, numerical values may be range limited, and domain names must adhere to standard specifications, etc.

- In most cases, validation is applied as you type and errors get highlighted.
- In a few cases, where consistency is required with other options, validation only happens upon submission to the server.

In any case, the M-Link server rejects invalid configuration updates. When that occurs, the page remains unsubmitted and an explanation pops up in a red, persistent box, which can be used to located the changes required to make a subsequent submission succeed.

An option may be mandatory if no default would be sensible. This is typical of list items that can be added, but not without filling in some essential properties. The mandatory

options are marked by a label reading “Required”. Also, the Submit button will remain disabled until all mandatory options are filled.

Due to dependencies between settings, it is possible for a change to enable mandatory settings on other pages. This somewhat rare condition is detected upon submission and produces a pop-up, named “Missing fields”, listing options that will be made visible by the update and which are currently not set, yet mandatory. These are editable in the pop-up and must be configured to complete the submission. It is recommended to make a note of the option paths listed on the “Missing fields” pop-up and visit the containing pages after the current submission completes successfully, as other setting changes may be desirable.

Other unrelated errors may occur, such as caused by a connection failure. They generate red, persistent pop-ups containing an explanatory message.

All error pop-ups can be dismissed by a simple click.

5.1.5 Special Options

5.1.5.1 Options Referring to Objects

Configuring an M-Link server generally requires importing external data objects which cannot be predetermined. For example, TLS features require keys and certificates which do not ship with the server. This is a two step process:

- Upload of a number of objects into M-Link managed stores.
- Selection of stored objects where needed in the configuration.

For convenience, it is in fact possible to upload an object from the places where an object selection is needed. But logically, the two operations are disconnected, as the same object may be selected in several places of the configuration. An object may also be uploaded but not selected anywhere (i.e. not in use in the configuration) or cease to be selected.

Object stores group objects of the same kind and act like internal catalogs. Setting a configuration option to an object from a store amounts to making a selection from a list of object names. Such options appear with an “Edit” button which offers a drop-down menu. The options in the drop-down depend on what is currently uploaded in the associated store.

Since some of the uploaded objects are themselves catalogs of items (such as security label catalogs), the internal stores can have two levels. The editing pop-up will then show two drop-down menus. The option is then rendered as a combination of the two choices.

The M-Link Web Console has a section for object store management. See Section 5.2.7, “Stores”.

5.1.5.2 List Options

There are basically two kinds of lists:

- Collections of configurable items.
- Options that happen to be multi valued.

The first case corresponds to M-Link Web Console sections (such as “Peer Controls”, see Section 5.2.2, “Peer Controls”) where the side bar menu leads to a page of cards representing the items in existence.

- Such pages may include an “Add item” button, unless displaying a fixed set of items.
- The cards may be clicked to access the features of the corresponding items.

- Items may also be reflected in the side bar menu, where a click will have the same effect.

There are variations with the second case of lists. Depending on complexity, the list items are either

- rendered in full or
- only partially but viewable one at a time in a pop-up.

In all cases, obvious controls are available to add, edit or delete items.

5.2 Configuration Sections

5.2.1 Global Options

This page features all the options that do not belong in any of the other sections. That is, options that apply server wide, excluding the following topics, which are substantial enough to require dedicated sections:

- TLS Configuration, with the exceptions mentioned below.
- Logging Configuration.
- Security Labelling Configuration.

The “Global Options” do include a couple of options that are TLS related:

Use HTTPS
: This controls whether the administrative interface (i.e. how the M-Link Web Console connects to the server) uses the HTTPS or HTTP protocols. Always use HTTPS, unless encryption is not available.

  If switching protocols is required, please note that the current connection will close, so the M-Link Web Console will appear to freeze and the browser must be told to visit the new location.

TLS Key Pair for Administrative HTTPS Interface
: This option is obviously required when HTTPS is enabled.

If the TLS Key Pair is misconfigured somehow, M-Link will automatically revert to expecting connections through HTTP.

Please note that the “Security Labelling Configuration” section is only available when option “Use Security Labelling” featured in the “Global Options” is set.

5.2.2 Peer Controls

This page collects all the Peer Controls currently configured, represented as cards. The card named “default” is special (it does not represent any real Peer), but is present here so it can be edited in the same way as proper Peer Controls.

- Peer Controls can be added by clicking the “Add item” button.
- Peer Control settings can be viewed or edited by clicking on the corresponding card.
- Deleting a Peer Control requires viewing it and using the “Delete...” button on the viewing page.

The “default” item will be used by M-Link when no Peer Control explicitly matches a domain. Please refer to Chapter 2, *Peer Controls and Links* for an explanation of how Peer Controls are used by the M-Link server.

5.2.3 Links

This page collects all the Links currently configured, represented as cards. Please refer to Chapter 2, *Peer Controls and Links* for an explanation of the concept.

- Links can be added by clicking the “Add item” button.
- Link settings can be viewed or edited by clicking on the corresponding card.
- Deleting a Link requires viewing it and using the “Delete...” button on the viewing page.

5.2.4 TLS Configuration

This page displays defaults and server wide options relating to TLS Configuration. See Chapter 3, *TLS Configuration* for details on this topic.

5.2.5 Logging Configuration

The side bar menu presents a “Logging Streams” entry, which leads to a page of cards.

- Logging Streams can be added by clicking the “Add item” button.
- Stream settings can be viewed or edited by clicking on the corresponding card.
- Deleting a Stream requires viewing it and using the “Delete...” button on the viewing page.

If the “Advanced Logging Level Configuration” is selected in a Logging Stream (and submitted), a “Facility Levels” entry appears in the side bar menu, which leads to a page of cards.

5.2.6 Security Labelling Configuration

The side bar menu entry for this page only exists if the global option “Use Security Labelling” is selected.

This page displays defaults and server wide options relating to Security Labelling. See Chapter 4, *Security Labels, Clearances and Policies* for details on this topic.

5.2.7 Stores

M-Link comes with a fixed set of object stores, each aggregating objects of a specific type and making them available as potential values for appropriate items in the configuration.

- The stores are represented here as cards.
- The contents of each store can be managed by clicking on the associated card.

Stores may be initially empty. They can be populated by clicking on the “Add item” button.

- In most cases, the operation involves choosing one, or a few, local files which contain the external embodiment of the object, for example, an XML file or a certificate in PEM format. The forms that pop up should be self-explanatory.
- In some cases, additional data needs to be input, such as a password.
- In all cases, a unique name must be given to the object. This defaults to the (main) file name without its extension, if any.

Store contents are presented as a list showing the names and upload dates of each object it contains.

- Each entry has a “Details” button leading to a read-only page showing significant details and all the properties of the object. The properties are specific to each type of object.

- This page also allows object deletion via the “Delete...” button, but only if the object is not in use, that is, has not been selected as the value of an item somewhere in the configuration. Otherwise, the header includes links to the places where it is used.

When objects are themselves a type of catalog, their sub structure is also presented in the form of a sub store. The item list of the store includes “Subitems” buttons, which lead to pages resembling the display of store contents. The differences derive logically from the fact that the sub items are parts of an immutable object:

- The list of sub items has no “Add item” button.
- The sub items cannot be deleted individually. Only the whole containing catalog object can be deleted.
- The sub items have neither upload dates nor source file names: these are attributes of the containing catalog object. They may have individual properties, however, since the containing catalog object may provide these.

5.2.8 Sessions

Refer to Chapter 6, *Monitoring* for further details.

The page displays a table showing details of all the sessions in existence. It includes two dynamic features on the far right:

- “End” buttons allow the administrator to terminate sessions.
- “Monitor” buttons open pop-ups which will display subsequent traffic going through the chosen session.

To refetch the current list of sessions use the refresh button.

5.3 Miscellaneous Features

5.3.1 Searching for Options

The side bar includes a search feature. This may help locate options by searching section titles, option names (technically titles) and values for text entered in the search box.

5.3.2 Server Version

The bottom of the side bar displays the current version of the server connected to the M-Link Web Console. Please include that version number in any support request.

5.3.3 About

The “About” button is located in the header bar and displays a side bar allowing the following:

- Display of the online manual,
- Display of the release notes,
- Display of product activation details and
- Management of the activation key (see Section 5.4, “Product Key Management”).

5.3.4 Profile

The “Profile” button is located in the header bar and displays the user name of the administrator currently logged in. It also includes two buttons:

“Logout”
Click this button to log out. The application returns to the “Login” page.

“Edit Credentials”
The main purpose of this button is to set a new password for the administrator currently logged in. It is also possible to edit the user name at the same time.

- Current (old) credentials are required input.
- Leaving field “New Username” blank preserves the current user name.

The changes take effect immediately (unless they are rejected). In other words, the user is not logged out, and opening a new connection to the server will require entering the new credentials.

5.4 Product Key Management

Product Key Management is accessible from the About side bar, which pops down when the “About” button is clicked in the top right corner. The side bar displays the details of the key currently loaded.

A new product key is required for the following purposes:

- To change the details of the activation.
- To extend the activation to a later release.
- To add or remove features.
- To widen or narrow the scope of the activation.

The “Update Key” button initiates the same operation required during initial activation.

- The pop-up may be closed at any time to continue working with the current key.
- The final step (the actual key update) overwrites the existing, but only if the new key is valid.

If an M-Link server is no longer needed, the product key can be removed using the “Deactivate” button. This operation cannot be undone, unless the key was saved elsewhere. As this returns the M-Link product to a non-activated state, product activation is then the only operation available.

Chapter 6 Monitoring

This chapter introduces the features available for monitoring the state of the server.

6.1 Remote Sessions Listing

To display a list of current sessions connecting the local server to remote servers, select the Sessions option in the main menu.

The list is a snapshot: use the refresh button provided on the page to get an updated list.

The list may be empty if there are no remote sessions. Otherwise, the list includes the following features for each session:

`SID`
: A unique identifier for the session.

`Type`
: A code for the session type. There are three types of sessions supported at the moment: `S2S`, `X2X`, `GCXP`. For details about these types see Section 2.2, “Peer Controls” and Section 2.6, “Links”.

`Dialback`
: When an incoming session authenticates using the dialback mechanism, a separate session is needed to interrogate the calling server according to DNS. If no session to the relevant server already exists, a new outgoing session is initiated which will be indicated in this column.

`TLS`
: This column indicates whether the session is protected with TLS.

`Authentications`
: This lists the XMPP domain endpoint pairs (local and remote) for which this session has been authenticated (allowing stanza routing). Specialised connection mechanisms authenticate based on Peer Control configuration and for those endpoint pairs will appear here once stanzas for that pairing have been exchanged. When Relaying is in use the 'local' domain of the pair will be the domain on to which M-Link will be forwarding the received stanzas, or on behalf of which it is forwarding.

`Initiated`
: This column indicates whether the local server (`Outbound`) or the remote server (`Inbound`) initiated the session.

`Direction`
: This column indicates whether the session is able to send (`Outgoing`) or receive (`Incoming`) stanzas, or both (`Bidirectional`).

`Local Port`
: Sessions are underpinned by a TCP/IP socket uniquely identified by IP address and port number. This column uses the standard IP notation (IPv4 or IPv6 as applicable) with port separated by a colon.

`Remote Port`
: As the Local Port, but relative to the remote server.

`Created On`
: The moment the session was created in the web browser's locale.

`Monitor`
: A button to access session related events. Clicking on the button for a particular session opens a pop-up and displays stream fragments (XML) carried by that

session since clicking. This is transient data which is lost when the window is closed. The window can also be reset by clicking its Clear button.

End
: The last column includes a button that allows termination of the session. Clicking on the button only displays a confirmation dialogue from which it is still possible to step back.

Chapter 7 HTTP API

This chapter covers the APIs available for programatic configuration of M-Link.

7.1 Authentication HTTP API

This section describes the HTTP REST end points for the management of administrator authentication.

7.1.1 Logging in as an Administrator

URL	/api/login
Method	POST
Request Content-Type	application/json

7.1.1.1 Description

The method allows user authentication based on a cleartext password.

7.1.1.2 Request

The request body contains values for username and password in the following JSON format:

```
{
    "username": "admin",
    "password": "adminpass"
}
```

7.1.1.3 Response

The response has no content. If successful, it includes a Set-Cookie header setting the loginToken cookie.

This cookie must be included in the Cookie header of all subsequent requests that require it, as specified in their related sections. Failure to include the loginToken cookie will fail these requests with a status code of 403 (FORBIDDEN) and no content in the response body.

7.1.1.4 Status Codes

204 (NO CONTENT)	Correct credentials were provided and cookie "loginToken" is set as HTTP only (so the browser environment does not have access to it).
403 (FORBIDDEN)	The credentials provided were incorrect.

7.1.2 Acquiring the Security Token

URL	/api/token
Method	GET
Response Content-Type	application/json

7.1.2.1 Description

Provides a CSRF token to authenticate users.

7.1.2.2 Request

The request has no content.

7.1.2.3 Response

If a valid login token is set, responds with 200 (OK) and a body containing the token as in the example below.

```
{
    "token": "..."
}
```

The client should store this token for later use in the X-Auth-Token header of requests that require it (please refer to their own sections). Failure to include this header with a valid token will fail these requests with a status code of 403 (FORBIDDEN) and no content in the response body.

Clients SHOULD fetch a new CSRF token after an hour, as they expire after two hours.

7.1.2.4 Authentication

The request must have a cookie obtained from the /api/login end point. Failure to provide a valid loginToken cookie will cause a return with an error status of 403 (FORBIDDEN) and an empty response body.

7.1.2.5 Status Codes

200 (OK)	A valid login token was set. The body contains: { "token": "..." }.
403 (FORBIDDEN)	No login token was present in the request or it was invalid.

7.1.3 Who Is Logged in?

URL	/api/me
Method	GET
Response Content-Type	application/json

7.1.3.1 Description

This end point returns the user name of the currently logged in administrator. The request must have a cookie obtained from /api/login.

7.1.3.2 Request

The response has no content.

7.1.3.3 Response

If no initial admin user is setup, returns with a body containing {}, otherwise:

```
{
    "username": "$username"
}
```

7.1.3.4 Authentication

The request must have a cookie obtained from the /api/login end point, if the initial admin user has been set up. Failure to provide a valid loginToken cookie then causes a return with an error status of 403 (FORBIDDEN) and an empty response body.

7.1.3.5 Status Codes

200 (OK)	If no initial admin user is set up, the request is trivially successful and the body contains: {}. If the initial admin user is set up, the login token cookie was present in the request and valid. The body contains: { "username": "$username" }
403 (FORBIDDEN)	The initial admin user is set up, but the login token cookie was missing or invalid. The body contains: {}

7.1.4 Changing Administrator username and/or password

URL	/api/change_credentials
Method	POST
Request Content-Type	application/json

7.1.4.1 Description

The method allows the administrative user's name or cleartext password to be changed. The current administrative username and password must be provided in addition to the desired new password and optionally new username.

7.1.4.2 Request

The request body contains values for existing and new username and password in the following JSON format:

```
{
    "username": "admin",
    "password": "adminpass",
    "newUsername": "administrator",
    "newPassword": "secret"
}
```

7.1.4.3 Response

The response has no content.

7.1.4.4 Authentication

The request must have a cookie obtained from the /api/login end point and a token obtained from the /api/token end point. Failure to comply will cause a return with an error status (such as 403 (FORBIDDEN)); please refer to the Authentication HTTP API for information on how to get the tokens.

7.1.4.5 Status Codes

204 (NO CONTENT)	Correct current credentials were provided and the new username and password have been stored.
422 (UNPROCESSABLE ENTITY)	The request body was invalid, or is missing required information, or the current credentials provided were incorrect.

7.2 Configuration HTTP API

This section describes the HTTP REST interface which allows third party clients to configure options in the option database.

The options can be thought of as held in a JSON document, described by a JSON Schema, which can be obtained from the /api/config/schema end point. The standard Schema has been extended with the following elements (please note: these are not always valid in combination):

```
{
    "properties": {
        "choice": {
            "isode_label": "someName",
            "isode_required": true,
            "isode_validator_type": "aValidatorName",
            "isode_multi_line": true,
            "isode_underlying_type": "NotNegativeExample",
            "isode_enum_titles": [
                "Display This Instead of the First Enum Value",
                "Display This Instead of the Second Enum Value"
            ],
            "isode_default_comes_from": "JSON Pointer",
            "isode_only_if": [
                {
                    "check": "is",
                    "ref": "1/enabled",
                    "value": true
                }
            ],
            "isode_store_uri": "/api/store/someObjectStore",
            "isode_substore_item": true,
            "isode_icon": "address-book"
        }
    }
}
```

isode_label
: This property is appropriate for lists of complex options (i.e. with nested sub options). It refers by name to one of the sub options, singling it out as distinctive of the list item. For example, the label might designate a sub option meant to be unique to each item and thus suitable as an item identifier. Lists of simple options do not require any label as they only carry one value per item.

isode_multi_line
: Indicates new-line characters in the option value are to be preserved.

isode_validator_type
: Denotes a validator which constrains string values to certain formatting rules. When taking input for an option with a validator, the GUI should validate it using end point /api/config/validator.

isode_underlying_type
: An informative property naming the underlying type of the option. It might be used to identify options that the end point consumer can validate by itself. However, it should only appear along side the "isode_validator_type" property.

isode_enum_titles
: Short definitions of the choices identified by an enum option. The option should also have the standard "enum" property, listing the enum values in the same order.

isode_only_if
: In the absence of any isode_only_if, the values of options fetched from the /api/config end point are either their default if unset, or the value they were last set to. An isode_only_if property is a boolean value computed as the logical AND of all the conditions it lists. The clauses in the isode_only_if connect the value of an option, referred to by a JSON Pointer (the "ref" element) to a constant value (the "value" element) with an operator (the "check" element) which is either "is" or "isNot". If the isode_only_if property is true, the behaviour is the same as when there is no isode_only_if property. If it is false, fetching the option value returns its default, whether or not the option was explicitly set. Please note that there are several ways to generate a default. If the option is complex (that is, it has nested sub options), a false isode_only_if property is logically applied to all the sub options, overriding any explicit "isode_only_if" on sub options. If the option is a list, a false "isode_only_if" makes it appear empty. While the "isode_only_if" is false, set values are not lost and options may still be set, modified or unset.

isode_default_comes_from
: Extends schema property default: instead of a value, refers to another option, via a JSON Pointer, from which to collect a default value. In cases where a default value is returned (e.g. fetching an unset option or fetching an option with false "isode_only_if"), "isode_default_comes_from" takes precedence over "default". Multiple options may be chained with "isode_default_comes_from". In other words, the target of an "isode_default_comes_from" may be unset or have a false "isode_only_if", in which case its own default is retrieved, wherever that comes from. Please note that some types have a natural default, meaning that even without any "isode_default_comes_from" or "default" a fetch will succeed. Other types must have some form of explicit default.

isode_required
: Indicates that a value must be provided when patching the configuration. Defaults to false if missing on an option. This is similar to the standard "required" property. However, this extension accommodates the "isode_only_if" extension: false "isode_only_if" conditions effectively disable the requirement to be set, as indeed only the default can then be fetched. In other words, unset options with "isode_required" but a false "isode_only_if" do conform to the schema (which would not be the case with a standard "required" property).

isode_store_uri
: Indicates that the option value is the unique identifier of an object held in an object store. Object stores are persistent containers of objects of a type specific to the store. These can be manipulated through a separate API (see section Object Stores HTTP API). The value of the isode_store_uri property is the path to the object store according to that API, relative to the server URI.

isode_substore_item
: If true, indicates that the option value is the unique identifier of an object within a substore. Substores are object stores which are themselves items of an object store. Please refer to section Object Stores HTTP API. The identifier only makes sense when knowing which substore it belongs to: Options with this property are expected to exist within a complex option also including a sub option with the same name plus "Substore" and providing the required information (see property isode_store_uri). Property isode_substore_item is incompatible with isode_underlying_type: Types with the latter are expected to support a validator (implicitly or explicitly), however substore items require validators which cannot be triggered via the /api/config/validator endpoint. They may still have validators applied by the server after calling PATCH /api/config, but those are not declared in the JSON Schema.

isode_icon
: The value is only a hint to the UI client, drawing meaning by convention: If it is appropriate for the option, the UI may interpret the value as the class name of an icon/glyph within some version of FontAwesome (tm) and use it where appropriate in relation to the rendering of that option.

Available operations are described below. Some end points allow multiple methods. They allow configuration retrieval, option update through JSON patches and validation. A status code of 200 (OK) implies that the request was successful.

7.2.1 Obtaining the JSON Schema

URL	/api/config/schema
Method	GET
Response Content-Type	application/json

7.2.1.1 Description

As mentioned in the overview of Section 7.2, “Configuration HTTP API”, the set of configuration options presents as a JSON document. In order to set values properly and derive defaults, it is necessary to retrieve a JSON Schema describing the whole document.

7.2.1.2 Request

Fetches the JSON Schema for the application options. The request body is empty.

7.2.1.3 Response

Returns the JSON Schema as a JSON document in the body.

Example:

```
{
    "$schema": "http://json - schema.org/draft - 04/schema#",
    "description": "Server configuration",
    "id": "http://isode.com/api/config/schema.json",
    "properties": {
        "useTls": {
            "default": true,
            "description": "Boolean value",
            "type": "boolean"
        },
        "max_stanza_size": {
            "default": 10024,
            "description": "Maximum size of a stanza",
            "maximum": 9223372036854775807,
            "minimum": -9223372036854775808,
            "type": "integer"
        }
    },
    "propertyOrder": [
        "max_stanza_size",
        "useTls"
    ],
    "title": "Config",
    "type": "object"
}
```

7.2.1.4 Authentication

The request must have a cookie obtained from the /api/login end point. Failure to comply will cause a return with an error status (such as 403 (FORBIDDEN)); please refer to the section relating to this end point.

7.2.1.5 Status Codes

Status codes relating specifically to this method:

200 (OK)	Schema successfully retrieved.
	Failure codes are possible, but only due to higher layer processing (such as authentication issues).

7.2.2 Retrieving the Current Configuration

URL	/api/config
Method	GET
Response Content-Type	application/json

7.2.2.1 Description

Retrieves the current configuration in JSON format. This configuration format is described in the JSON Schema provided by the /api/config/schema end point. Please note that the document may be incomplete: where options have not been explicitly set they are omitted, as their values can be derived either from the default included in the schema, or from an isode_default_comes_from, or the type has a natural default, such as strings which can be empty.

7.2.2.2 Request

Fetches the current configuration in JSON format. The request body is empty.

7.2.2.3 Response

The JSON document returned in the body conforms to the JSON Schema. It only includes explicitly set options.

Example:

```
{
    "max_stanza_size": 10000
}
```

7.2.2.4 Authentication

The request must have a cookie obtained from the /api/login end point. Failure to comply will cause a return with an error status (such as 403 (FORBIDDEN)); please refer to the section relating to this end point.

7.2.2.5 Status Codes

Status codes relating specifically to this method:

200 (OK)	Successfully retrieved the configuration.
	Failure codes are possible, but only due to higher layer processing (such as authentication issues).

7.2.3 Applying Changes to the Configuration

URL	/api/config
Method	PATCH
Request Content-Type	application/json-patch+json
Response Content-Type	application/json

7.2.3.1 Description

Applies changes, described in a JSON Patch document, to the current configuration.

7.2.3.2 Request

The request body should contain a valid JSON Patch document describing the changes to be applied to the configuration, seen as a JSON document conforming to the JSON Schema provided by the /api/config/schema end point. A JSON Patch is an array of objects, each describing a single operation to be performed on a specific option referred to by its path (a JSON Pointer).

Only a sub set of JSON Patch operations are supported: replace, add, remove.

replace
: This operation sets an existing single option to a value. It fails if the path leads to a complex or list option.

  Example:

```
[
    {
        "op": "replace",
        "path": "/max_stanza_size",
        "value": 99000
    }
]
```

add
: This operation adds an item to a list. The path must lead to a list option with '/-' appended to it (suggesting the index number beyond the current last). If the item is complex, the value is a compound JSON object matching the sub tree. Only list items can be added as the option tree is fixed.

  Example: N.B. the list assumed below does not feature in the simple schema example given out above:

```
[
    {
        "op": "add",
        "path": "/imDomains/0/users/-",
        "value": {
            "jid": "alice@wonderland.lit",
            "password": "secret"
        }
    }
]
```

remove
: This operation removes an item from a list. The path must lead to an existing list item. Other options are not removable.

  Example: N.B. the list assumed below does not feature in the simple schema example given out above:

```
[
    {
        "op": "remove",
        "path": "/imDomains/0/users/5"
    }
]
```

7.2.3.3 Response

If the patch was successfully applied, the new configuration is returned in the response body in the same way as in response to a GET /api/config.

Example:

```
{
    "max_stanza_size": 99000,
    "useTls": true
}
```

If the patch failed, the body is empty.

7.2.3.4 Authentication

The request must have a cookie obtained from the /api/login end point and a token obtained from the /api/token end point. Failure to comply will cause a return with an error status (such as 403 (FORBIDDEN)); please refer to the sections relating to these end points.

7.2.3.5 Status Codes

Status codes relating specifically to this method:

200 (OK)	The patch was successfully applied.
415 (UNSUPPORTED MEDIA TYPE)	The content type or the JSON syntax was invalid.
422 (UNPROCESSABLE ENTITY)	The JSON Patch provided was invalid or could not be applied.
	Other failure codes are possible, but only due to higher layer processing (such as authentication issues).

7.2.4 Validating Option Values

URL	/api/config/validator
Method	POST
Request Content-Type	application/json
Response Content-Type	application/json

7.2.4.1 Description

This end point provides a facility to validate a particular option value before submitting it to end point /api/config with PATCH. Most value types are fully defined in the JSON Schema. For example, integers might have a maximum and minimum. Other types are defined as strings but are actually more elaborate and/or have specific formatting restrictions that cannot be easily expressed in the schema. These will have a "validator_type" property with which to query the option database through this end point.

7.2.4.2 Request

The body is a JSON document defining a type and value to validate. The type is the value of a "validator_type" property and should be recognized by the option database.

Example:

```
{
    "type": "typeName",
```

```
    "value": "valueToTest"
}
```

7.2.4.3 Response

If the validator ("type" in the request) is supported and the value valid, the body contains:

```
{
    "valid": true
}
```

If the validator is supported but the value invalid, the body content resembles the following, the value of userMessage being supplied by the validator.

```
{
    "valid": false,
    "userMessage": "a hint"
}
```

7.2.4.4 Authentication

The request must have a cookie obtained from the /api/login end point. Failure to comply will cause a return with an error status (such as 403 (FORBIDDEN)); please refer to the section relating to this end point.

7.2.4.5 Status Codes

Status codes relating specifically to this method:

200 (OK)	The validator is supported. If the value is valid, the body contains: { "valid": true }, otherwise: { "valid": false, "userMessage": "a hint" }
400 (BAD REQUEST)	Unsupported content type or unsupported validator. In the latter case, the body contains: { "error": "message" }
	Other failure codes are possible, but only due to higher layer processing (such as authentication issues).

7.3 Object Stores HTTP API

This section describes the HTTP end points for the management of Object Stores. All Object Store end points require a valid login token. The calls that modify the store or its contents will additionally require a CSRF token.

The API provides access to two kinds of stores: Object Stores and Object Substores. Object Substores provide another hierarchy layer to Object Stores where each item in the Object Store is another Object Store containing items - e.g. every Security Label Catalog in the Security Label Catalog Store can be addressed as a substore, providing access to all the labels in the catalog. Substores are immutable, so any operation attempting to modify a substore will fail.

Each available Store has a base path associated with it. Store paths will be represented by */`storepath`* in this section. The base paths for most stores can be found from the *GET /api/stores* endpoint.

The base paths for substores are generated by taking the path of the base store, */storepath*, appending the id of the item in the base store that you want to use as a substore, and appending */sub_store*. For example, if a Security Label Catalog store was located at */api/stores/label_catalogs*, and contained a catalog with id 'a64ef', */api/stores/label_catalogs/a64ef/sub_store* would be the base path for a substore containing the labels provided by that catalog.

7.3.1 List available Object Stores

URL	/api/stores
Method	GET
Response Content-Type	application/json

7.3.1.1 Description

This method can be used to discover the object stores that are available.

7.3.1.2 Request

The request has no content.

7.3.1.3 Response

Returns an array of objects. Each object has at least the members `name`, `path`, `description`, and `upload_fields` as shown. The `path` member represents the base path of each store. If the `has_substores` member is present and set to true, this store has substores. The `upload_fields` member is an array of specifications of the fields which the store requires when creating a new entry. In the simple case, this is a single file containing the data of the entry, but more complex stores may require multiple components to be supplied (e.g. a certificate chain, private key and passphrase for a TLS identity).

```
[
    {
        "name": "Example Store",
        "path": "/api/stores/example",
        "description": "All available examples are stored here.",
        "upload_fields": [
            {
                "name": "object",
                "type": "file",
                "title": "Filename",
                "description": "Choose a file to upload",
                "accept": ".pem,.crt",
                "required": true
            }
        ]
    },
    {
        "name": "Example Store Containing Substores",
        "path": "/api/stores/example_catalogs",
        "description": "Example catalogs are stored here.",
        "has_substores": true
    }
]
```

7.3.1.4 Authentication

The request must have a cookie obtained from the /api/login end point. Failure to comply will cause a return with an error status (such as 403 (FORBIDDEN)); please refer to the Authentication HTTP API for information on how to get the token.

7.3.1.5 Status Codes

200 (OK)	The request was successful.
403 (FORBIDDEN)	The login token cookie was missing or invalid.

7.3.2 List all objects in a Store or Substore

URL	/*storepath*
Method	GET
Response Content-Type	application/json

7.3.2.1 Description

This call returns a list of all items in the specified Object Store or Object Substore.

7.3.2.2 Request

The request has no content.

7.3.2.3 Response

Returns an array of objects. Each object consists of the members `id`, `name` and an optional `creationDate`, in seconds since 1970-01-01T00:00:00Z. `name` can be used for display purposes. The `id` of a given object will not change, and is used in methods that operate on a specific stored item.

```
[
    {
        "id": "7b5237b4-c3ea-4f9f-8d16-20eb96486b0e",
        "name": "An example object",
        "creationDate": 1582798994
    },
    {
        "id": "174edbfa-e2b3-4e5c-b5bb-d3783a364e3d",
        "name": "Another object",
        "creationDate": 1585582903
    }
]
```

7.3.2.4 Authentication

The request must have a cookie obtained from the /api/login end point. Failure to comply will cause a return with an error status (such as 403 (FORBIDDEN)); please refer to the Authentication HTTP API for information on how to get the token.

7.3.2.5 Status Codes

200 (OK)	The request was successful.
403 (FORBIDDEN)	The login token cookie was missing or invalid.
404 (NOT FOUND)	The requested store does not exist.

7.3.3 Add an object to the Store

URL	/*storepath*
Method	POST
Request Content-Type	multipart/form-data
Response Content-Type	javascript/json

7.3.3.1 Description

This call can be used to add an object to the specified store.

The object content is wrapped within a multipart/form-data, as explained in the request section below. Its *`filename`* parameter is meant to record the name of the file the object is uploaded from. The name of the new object in the store may also be set using a `name` field, but defaults to the filename. It is returned in the response as the value of property "name", along with property "id" which is generated by the POST operation as a unique identifier. Other attributes may also exist, but those are object type specific.

7.3.3.2 Request

The call expects a valid multipart/form-data payload.

You can provide form values for each field mentioned in the `upload_fields` entry from the HTTP GET which discovered the Object Store. Fields marked as required must always be sent.

Fields of type `file` must have their *`filename`* attribute set. When multiple fields of type `file` are submitted, the object will be stored using the first filename encountered.

This is compatible with regular HTTP file upload mechanisms found in most web clients. Multiple objects may thus be stored in a single request. Objects should a Content-Type values of application/octet-stream, or something more specific.

The multipart/form-data payload may also include a field named `name`. Its value is an arbitrary name to be associated with the object. It must have a Content-Type of text/plain with charset of UTF-8, but these are default values. If omitted, the *`filename`* parameter of the first field which contains one will be used.

7.3.3.3 Response

If the call succeeds, the same object information as returned by 'Get object information'. In case of an error, the body will be an object with a `message` member with a user-presentable error.

7.3.3.4 Authentication

The request must have a cookie obtained from the /api/login end point and a token obtained from the /api/token end point. Failure to comply will cause a return with an error status (such as 403 (FORBIDDEN)); please refer to the Authentication HTTP API for information on how to get the tokens.

7.3.3.5 Status Codes

200 (OK)	The request was successful.
403 (FORBIDDEN)	The login token cookie was missing or invalid.
404 (NOT FOUND)	The requested store does not exist.
422 (UNPROCESSABLE ENTITY)	The request body was invalid, or is missing required information.

7.3.4 Retrieve an object from the Store

URL	/*`storepath`*/*`id`*
Method	GET
Response Content-Type	*Content specific*

7.3.4.1 Description

This call returns the object data for an item in an Object Store. Unless the store holds composite objects which are not convertible to a single file, in which case this feature is disabled and a retrieval attempt yields an error.

7.3.4.2 Request

The request has no content.

7.3.4.3 Response

Returns the data associated with object *id* in the specified store. The `Content-Type` will depend on the object stored. In case of an error, the Content-Type will be `application/json`, and the body will be an object with a `message` member with a user-presentable error.

7.3.4.4 Authentication

The request must have a cookie obtained from the /api/login end point. Failure to comply will cause a return with an error status (such as 403 (FORBIDDEN)); please refer to the Authentication HTTP API for information on how to get the token.

7.3.4.5 Status Codes

200 (OK)	The request was successful.
403 (FORBIDDEN)	The login token cookie was missing or invalid.
404 (NOT FOUND)	The requested store or item does not exist or the store does not support content retrieval.

7.3.5 Rename an object

URL	/*storepath*/*id*
Method	POST
Request Content-Type	javascript/json
Response Content-Type	javascript/json

7.3.5.1 Description

This call updates the name of the specified object in the Object Store.

7.3.5.2 Request

The request body must be an object with a single *name* member that holds the desired new name of the object.

```
{
    "name": "New object name"
}
```

7.3.5.3 Response

If the call succeeds, the object identified by *id* will have its *name* updated. In case of an error, will return an object with a `message` member with a user-presentable error.

Please note that the call will fail if the object is read-only (see section Get object information).

7.3.5.4 Authentication

The request must have a cookie obtained from the /api/login end point and a token obtained from the /api/token end point. Failure to comply will cause a return with an error status (such as 403 (FORBIDDEN)); please refer to the Authentication HTTP API for information on how to get the tokens.

7.3.5.5 Status Codes

204 (NO CONTENT)	The request was successful.
403 (FORBIDDEN)	The login token cookie was missing or invalid.
404 (NOT FOUND)	The requested store or item does not exist.
422 (UNPROCESSABLE ENTITY)	The request body was invalid, is missing the required information, or refers to a read-only object.

7.3.6 Remove an object from the Store

URL	/*storepath*/*id*
Method	DELETE
Response Content-Type	javascript/json

7.3.6.1 Description

This call removes the specified object from the Object Store.

7.3.6.2 Request

The request has no content.

7.3.6.3 Response

If the call succeeds, the object identified by *id* will no longer be present in the Object Store. In case of an error, will return an object with a `message` member with a user-presentable error.

Please note that the call will fail if the object is read-only (see section Get object information).

7.3.6.4 Authentication

The request must have a cookie obtained from the /api/login end point and a token obtained from the /api/token end point. Failure to comply will cause a return with an error status (such as 403 (FORBIDDEN)); please refer to the Authentication HTTP API for information on how to get the tokens.

7.3.6.5 Status Codes

204 (NO CONTENT)	The request was successful.
403 (FORBIDDEN)	The login token cookie was missing or invalid.
404 (NOT FOUND)	The requested store or item does not exist.
422 (UNPROCESSABLE ENTITY)	The requested item is still in use in an active configuration, or is read-only.

7.3.7 Get object information

URL	/*storepath*/*id*/info
Method	GET
Response Content-Type	application/json

7.3.7.1 Description

This call returns all available additional information on an object in the specified store.

7.3.7.2 Request

The request has no content.

7.3.7.3 Response

Returns the information as an object with at least the following members: `id`, `name`.

Further there may be the following members: `creationDate`, `filename`, `contentType`, `readOnly`.

`id`
: The object's unique identifier generated when it was added to the store.

`name`
: The name chosen for the object. This may have defaulted to the filename of origin, or may have been set when adding the object or using the rename procedure.

`filename`
: The filename given when adding the object.

`contentType`
: The MIME content type given when adding the object.

`creationDate`
: The date and time the object was added. This is conveyed as a number of seconds since the Epoch (1970-01-01T00:00:00Z).

`readOnly`
: If this member is present and set to `true`, the object is read-only and cannot be removed, renamed or overwritten.

Additionally, there may be an `attributes` member with an object as value. This object will hold additional information; which attributes are available depends on the store. Attributes which have empty values will not be returned, and this may mean that the `attributes` member is empty.

```
{
    "id": "7b5237b4-c3ea-4f9f-8d16-20eb96486b0e",
    "name": "An example object",
    "filename": "example.xml",
    "contentType": "text/xml",
    "creationDate": 1574091301,
    "attributes": {
        "valid_from": "2020-01-01Z00:00:00"
    }
}
```

7.3.7.4 Authentication

The request must have a cookie obtained from the /api/login end point. Failure to comply will cause a return with an error status (such as 403 (FORBIDDEN)); please refer to the Authentication HTTP API for information on how to get the token.

7.3.7.5 Status Codes

200 (OK)	The request was successful.
403 (FORBIDDEN)	The login token cookie was missing or invalid.
404 (NOT FOUND)	The requested store or item does not exist.

Appendix A Pre-defined Transformations

This appendix describes the Transformations that are included with M-Link.

A.1 Block common File Transfer and VOIP mechanisms

This transformation will ensure that common mechanisms for file transfer and VOIP will be removed.

The protocols covered are In-Band Bytestreams (*XEP-0047*), Stream Initiation (*XEP-0095*), and Jingle (*XEP-0166*) and their associated standards.

When using this transformation, it is recommended to set the *Action on Blocked Stanza* option on the rule to *Bounce* to ensure the blocked IQ requests are replied to with an error.

A.2 Normalize XML for M-Guard

This transformation should be used to normalize the XML sent to M-Guard servers. It should generally be the last one configured on a given peer.

This will ensure `xml:lang` attributes are all lowercase, and that whitespace is normalized.

A.3 Strip Chat State Notifications

This transformation will remove Chat State Notifications (*XEP-0085*) elements from message stanzas. Chat State Notifications are used to inform users of the state of their conversation partner in a chat session ("inactive", "composing", "paused", etc.)

Disabling these notifications may be desirable in cases where bandwidth is restricted.

A.4 Strip Message Delivery Receipts

This transformation removes Message Delivery Receipts (*XEP-0184*) elements from message stanzas. Message Delivery Receipts are a mechanism that will allow a sender to ask the recipient to acknowledge receipt of a content message by returning an ack message.

Disabling Message Delivery Receipts may be desirable in cases where bandwidth is restricted.

A.5 Strip XHTML-IM parts from messages

This transformation will remove XHTML-IM (*XEP-0071*) markup from message stanzas. XHTML-IM is a method to include lightweight text markup.

Disabling XHTML-IM parts may be desirable in cases where bandwidth is restricted.

Appendix B Standards Supported by M-Link

An overview of Open Standards M-Link products conform to.

B.1 Supported by M-Link Edge

RFC 1035 Domain names - implementation and specification

RFC 6120 Extensible Messaging and Presence Protocol (XMPP): Core

RFC 6121 Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence

RFC 6122 Extensible Messaging and Presence Protocol (XMPP): Address Format

XEP-0004 Data Forms

XEP-0030 Service Discovery

XEP-0082 XMPP Date and Time Profiles

XEP-0198 Stream Management

M-Link only supports Stanza Acknowledgements. Stream Resumption is not supported.

XEP-0199 XMPP Ping

XEP-0220 Server Dialback

XEP-0258 Security Labels in XMPP

XEP-0361 Zero Handshake Server to Server Protocol

Appendix C Glossary

This appendix provides a glossary of terms.

Technical Terms

Domain
See Domain Name.

Domain Name
A name within the *Domain Name System*. See [RFC1035].

Domain Name System (DNS)
A service for providing a mapping between *domain names* (for example, `example.com`) and *IP addresses*. See *RFC 1035*.

Extensible Messaging and Presence Protocol (XMPP)
A collection of open standards for real-time communication, including those for *instant messaging*, presence, and *multi-user chat*. See [RFC6120].

Extensible Markup Language (XML)
A markup language used to represent structured information which is designed to be readable by both humans and machines.

Fully Qualified Domain Name (FQDN)
A complete *domain name* identifying a host system or other entity on the Internet. See Also Domain Name System (DNS).

Guard Content Exchange Protocol (GCXP)
Protocol used for communications between M-Link and M-Guard.

Instant Messaging (IM)
Real-time text-based chatting between two or more people. XMPP IM is described in [RFC6121].

Multi-User Chat (MUC)
An *instant messaging* service allowing multiple users to chat with each other; a group chat service. See [XEP-0045].

IP address
An address which identifies a host machine on an Internet network. For IPv4, it is 32-bit number commonly written in dotted number notation of the form `192.0.1.100`. For IPv6, it is a 128-bit number commonly written in a notation of the form `2001:db8::100`.

Appendix D References

The documents listed in this appendix provide references to the appropriate standards and other sources of information.

If documents can be obtained electronically, the location is stated as part of the reference. For other documents, please see Section D.6, “Obtaining documents”.

D.1 RFCs

RFC 1035
Domain names - implementation and specification [https://tools.ietf.org/html/rfc1035]. P.V. Mockapetris, November 1987

RFC 6120
Extensible Messaging and Presence Protocol (XMPP): Core [https://tools.ietf.org/html/rfc6120]. P. Saint-Andre, March 2011

RFC 6121
Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence [https://tools.ietf.org/html/rfc6121]. P. Saint-Andre, March 2011

RFC 6122
Extensible Messaging and Presence Protocol (XMPP): Address Format [https://tools.ietf.org/html/rfc6122]. P. Saint-Andre, March 2011

RFC 6901
JavaScript Object Notation (JSON) Pointer [https://tools.ietf.org/html/rfc6901]. P. Bryan, K. Zyp, M. Nottingham, April 2013

RFC 6902
JavaScript Object Notation (JSON) Patch [https://tools.ietf.org/html/rfc6902]. P. Bryan, M. Nottingham, April 2013

RFC 7578
Returning Values from Forms: multipart/form-data [https://tools.ietf.org/html/rfc7578]. L. Masinter, July 2015

RFC 8259
The JavaScript Object Notation (JSON) Data Interchange Format [https://tools.ietf.org/html/rfc8259]. T. Bray, December 2017

D.2 XMPP Extension Protocols

XEP-0004
Data Forms [https://xmpp.org/extensions/xep-0004.html]. Ryan Eatmon, Joe Hildebrand, Jeremie Miller, Thomas Muldowney, Peter Saint-Andre, June 2021

XEP-0030
Service Discovery [https://xmpp.org/extensions/xep-0030.html]. Joe Hildebrand, Peter Millard, Ryan Eatmon, Peter Saint-Andre, October 2017

XEP-0045
Multi-User Chat [https://xmpp.org/extensions/xep-0045.html]. Peter Saint-Andre, March 2021

XEP-0047
In-Band Bytestreams [https://xmpp.org/extensions/xep-0047.html]. Justin Karneges, Peter Saint-Andre, January 2021

XEP-0049
Private XML Storage [https://xmpp.org/extensions/xep-0049.html]. Peter Saint-Andre, Russell Davis, March 2004

XEP-0054
vcard-temp [https://xmpp.org/extensions/xep-0054.html]. Peter Saint-Andre, July 2008

XEP-0071
XHTML-IM [https://xmpp.org/extensions/xep-0071.html]. Peter Saint-Andre, March 2018

XEP-0082
XMPP Date and Time Profiles [https://xmpp.org/extensions/xep-0082.html]. Peter Saint-Andre, Tobias Markmann, September 2013

XEP-0085
Chat State Notifications [https://xmpp.org/extensions/xep-0085.html]. Peter Saint-Andre, Dave Smith, September 2009

XEP-0092
Software Version [https://xmpp.org/extensions/xep-0092.html]. Peter Saint-Andre, February 2007

XEP-0095
Stream Initiation [https://xmpp.org/extensions/xep-0095.html]. Thomas Muldowney, Matthew Miller, Ryan Eatmon, November 2017

XEP-0166
Jingle [https://xmpp.org/extensions/xep-0166.html]. Scott Ludwig, Joe Beda, Peter Saint-Andre, Robert McQueen, Sean Egan, Joe Hildebrand, September 2018

XEP-0184
Message Delivery Receipts [https://xmpp.org/extensions/xep-0184.html]. Peter Saint-Andre, Joe Hildebrand, August 2018

XEP-0191
Blocking Command [https://xmpp.org/extensions/xep-0191.html]. Peter Saint-Andre, March 2015

XEP-0198
Stream Management [https://xmpp.org/extensions/xep-0198.html]. Justin Karneges, Peter Saint-Andre, Joe Hildebrand, Fabio Forno, Dave Cridland, Matthew Wild, July 2018

XEP-0199
XMPP Ping [https://xmpp.org/extensions/xep-0199.html]. Peter Saint-Andre, March 2019

XEP-0220
Server Dialback [https://xmpp.org/extensions/xep-0220.html]. Jeremie Miller, Peter Saint-Andre, Philipp Hancke, March 2015

XEP-0227
Portable Import/Export Format for XMPP-IM Servers [https://xmpp.org/extensions/xep-0227.html]. Magnus Henoch, Waqas Hussain, March 2010

XEP-0258
Security Labels in XMPP [https://xmpp.org/extensions/xep-0258.html]. Kurt Zeilenga, November 2018

XEP-0280
Message Carbons [https://xmpp.org/extensions/xep-0280.html]. Joe Hildebrand, Matthew Miller, Georg Lukas, May 2021

XEP-0361
Zero Handshake Server to Server Protocol [https://xmpp.org/extensions/xep-0361.html]. Steve Kille, September 2017

D.3 W3C Recommendations

XSLT 1.0

XSL Transformations (XSLT) Version 1.0 [https://www.w3.org/TR/xslt-10/]. W3C Recommendation, 16 November 1999

D.4 Other Publications

JSON Schema [https://json-schema.org/].

Open XML SPIF [http://www.xmlspif.org/].

D.5 Isode White Papers

Security Policy Based Access Controls [https://www.isode.com/products/security-policy-infrastructure.html].

Using Security Labels to Control Message Flow in XMPP Services [https://www.isode.com/whitepapers/controlling-message-flow.html].

D.6 Obtaining documents

D.6.1 ISO/IEC documents

ISO/IEC standards and draft documents may be obtained from:

ISO Central Secretariat
International Organization for Standardization (ISO)
1, rue de Varembé
Case postale 56
CH-1211 Geneva 20
Switzerland

Telephone: +41 22 749 01 11

Fax: +41 22 733 34 30

Web: http://www.iso.org/

D.6.2 ITU-T (CCITT) documents

International Telecommunications Union
Place des Nations
CH-1211 Geneva 20
Switzerland

Telephone: +41 22 730 61 41 Fax: +41 22 730 51 94

Email: sales@itu.int

Web: http://www.itu.int/

D.6.3 RFCs

Electronic copies of RFCs are available from the following servers:

- http://ftp.isi.edu/in-notes/
- http://www.rfc-editor.org/