TigaseDoc

发行版本 *0.1*

Tigase, Inc.

2022 年 08 月 19 日

管理员指南

欢迎使用 Tigase HTTP API 用户指南。HTTP API 允许您直接从浏览器使用简单易用的界面管理、配置、聊天和向 Tigase 服务器发送命令！我们将指导您完成设置、运行和了解 HTTP API 的一些功能。

设置和配置

默认情况下，Tigase 将启动并运行 http servlet。

Tigase HTTP API 组件是一个通用容器，用于提供其他 HTTP 相关功能作为模块。它默认配置为以 http 的名称运行。即使未配置，Tigase XMPP 服务器的安装也会默认以相同的名称启用此组件。

CHAPTER 1

HTTP API 组件

Tigase HTTP API 组件是一个通用容器，用于提供其他 HTTP 相关功能作为模块。它默认配置为以 http 的名称运行。即使未配置，Tigase XMPP 服务器的安装也会默认以相同的名称启用此组件。

1.1 Tigase HTTP-API 发行说明

1.1.1 Tigase HTTP-API 2.2.0 发行说明

欢迎使用 Tigase HTTP-API 2.2.0！这是一个包含许多修复和更新的功能版本

主要变化

- 默认情况下启用 HTTP 文件上传，带有额外的可选 AWS S3 兼容后端
- 对 Web 设置的改进，使安装更加简单
- 允许在根上下文中公开 `.well-known` 以促进 XEP-0156: Discovering Alternative XMPP Connection Methods
- 添加选项以将请求从 http 重定向到 https

所有更改

- #http-65: 更详细的日志
- #http-86: 为 http-upload 添加 s3 后端
- #http-91: 功能屏幕上设置中的项目未对齐
- #http-93: 更新网络安装程序文档
- #http-95: 默认启用 HTTP 文件上传
- #http-96: 启用集群模式/ACS 不会将其添加到生成的配置文件中
- #http-98: 自九月以来，设置测试失败
- #http-99: 强制执行最大文件容量限制
- #http-100: 阻止启用所有 Message* 插件
- #http-101: 阻止启用所有 Mobile* 插件
- #http-102: 最后一个活动插件的处理应该得到改进
- #http-103: 启用 http-upload 应该提供有关设置域/存储要求的信息
- #http-105: 处理文件名中的禁用字符
- #http-106: 无法删除不存在的 VHost 的用户
- #http-107: 允许在根上下文中公开 `.well-known`
- #http-108: 添加选项以将请求从 http 重定向到 https
- #http-109: 将组件迁移到 TK 后缺少 openAccess 选项
- #http-110: 添加对查询和管理上传文件的支持
- #http-111: DefaultLogic.removeExpired 删除槽失败
- #http-113: 仅当 X-Forwarded-Proto 具有特定值时才添加条件重定向
- #http-114: TigaseDBException：无法分配插槽
- #http-116: 限制 VHosts 列表不适用于基于 JDK 的 http-server
- #http-117: Http 重定向在 docker 中不起作用
- #http-119: 无法通过 Admin WebUI 更改 VHost 配置
- #http-120: 改进 S3 对 HTTP 文件上传的支持，以接受 S3 存储配置的自定义 URL 和凭据
- #http-121: 弃用 DnsWebService 并重写 /.well-known/host-meta 生成器

1.2 可用模块

1.2.1 管理界面模块

这是使用 HTTP 浏览器管理 Tigase XMPP 服务器的非常简单的模块。它允许管理员从 HTTP 浏览器执行临时命令，允许在运行时更改一些配置选项。可以通过将浏览器指向 http://server.address:8080/admin/ 并使用管理员凭据登录来访问它。

1.2.2 索引模块

该模块默认部署在 / ，并在请求时为虚拟主机提供已安装和可用模块的列表。

1.2.3 REST 模块

该模块提供类似 REST 的 API 来访问 Tigase XMPP 服务器。它使用 Groovy 脚本来处理 HTTP 请求并准备响应。

1.2.4 服务器状态模块

> **警告：** 该模块仍在进行中！

该模块旨在呈现当前服务器状态并报告可能的问题。

1.2.5 设置模块

创建模块以作为 Tigase XMPP 服务器的基于 Web 的安装程序和配置实用程序。允许您修改基本的 Tigase XMPP 服务器设置，也就是说，与数据库访问有关。更改可以从此模块保存到配置文件。

1.2.6 网页界面模块

该模块包含基于 Tigase JaXMPP 客户端库的完整网络客户端，允许用户聊天、管理联系人列表（名册）、浏览消息存档等。有关此模块的更多信息，请参阅 管理指南。

1.2.7 DNS 网络服务模块

对于基于网页的 XMPP 客户端，无法执行 DNS SRV 请求来查找特定域的 XMPP 服务器托管地址。为了解决这个问题，创建了 DNS 网络服务模块。

它处理传入的 HTTP GET 请求并使用 `Host` HTTP 标头执行 DNS 请求，如 XEP-0156: 发现替代 XMPP 连接方法 中指定的那样。结果以前面提到的 XEP 中指定的 XML 或 JSON 格式返回。

默认情况下，它部署在 `dns-webservice` 并且 XML 响应的路径是 `/dns-webservice/.well-known/host-meta`，JSON 响应的路径是 `/dns-webservice/-known/host-meta.json`。

1.2.8 用户状态端点模块

此模块被设计为 REST API 用户状态正常工作所需的端点。它无法使用 HTTP/REST API 访问，因此它可以（并且在大多数情况下应该）处于活动状态。

1.3 常用模块配置

1.3.1 启用/禁用模块

每个模块都可以通过以下方式调整其活动来激活或禁用：

```
http {
    %module_id% (active: false) {}
}
```

备注： 您需要将 `%module_id%` 替换为想要更改活动的模块的 ID（在这种情况下，它将禁用模块）。

禁用 REST 模块。

```
http {
    rest (active: false) {}
}
```

1.3.2 上下文路径

此属性允许您更改模块使用的上下文路径。换句话说，它允许您更改模块使用的前缀。默认情况下，每个模块（索引模块除外）都使用与模块 ID 相同的上下文路径。例如，REST 模块 ID 导致上下文路径 `/rest`

将 REST 模块的上下文路径更改为 `/api`。

```
http {
    rest {
        context-path = '/api'
    }
}
```

1.3.3 虚拟主机列表

这提供了将模块限制为仅在列出的虚拟主机上可用的能力，并允许将上下文路径设置为 / 并用于多个模块。属性接受字符串列表，在 config.tdsl 文件格式的情况下是逗号分隔的域名列表，在 DSL 中它被写为字符串列表（参见*Complex Example*）。

将 REST 模块移动为仅可用于定向到 `api.example.com` **的请求**。

```
http {
    rest {
        vhosts = [ 'api.example.com' ]
    }
}
```

1.3.4 复杂的例子

在这个例子中，我们将禁用 Index 模块并将 REST 模块移动到 `http://api.example.com/` 和 `http://rest.example.com`。

```
http {
    index (active: false) {}
    rest {
        context-path = '/'
        vhosts = [ 'api.example.com', 'rest.example.com' ]
    }
}
```

1.4 模块特定配置

Tigase 将尝试在端口 8080 启动独立的 Jetty HTTP 服务器并启动默认模块，包括 `RestModule`，它将在 /rest 路径中添加 REST API 的上下文。`RestModule` 还将加载位于 `scripts/rest/*` 目录中的所有常规脚本，并适当操作将它们绑定到 `/rest/*` 路径。

注意： 处理 HTTP 请求的脚本可在 `src/scriopts/groovy/tigase/rest/` 目录中的组件存储库中找到。

Tigase 的 REST 组件带有两个可以单独启用、禁用和配置的模块。组件属性的模块的通用设置以下列格式使用：`component_name (module: value) {}`，以下设置可用于两个列出的模块：

- `active` - 布尔值 true/false 以启用或禁用模块。
- `context-path` - 模块应该可用的 HTTP 上下文的路径。
- `vhosts` - 模块应该可用的虚拟主机的逗号分隔列表。如果未配置，该模块将可用于所有虚拟主机。

1.4.1 Rest 模块

这是为 REST API 提供支持的模块。可用属性：

- `rest-scripts-dir` - 如果您不希望使用默认值（脚本/休息），则可以指定处理 REST 请求的脚本路径。

API 密钥

在以前的版本中，可以使用配置文件中的条目为 REST 模块配置 `api-keys`。在最近的版本中，我们决定删除此配置选项。现在，默认情况下 Tigase XMPP 服务器要求将 API 密钥传递给所有请求，您需要先配置它们，然后才能使用 REST API。

相反，您应该使用 REST 模块 JID 上可用的 ad-hoc 来：

- 添加 API 密钥（`api-key-add`）
- 更新 API 密钥（`api-key-update`）
- 删除 API 密钥（`api-key-remove`）；

小技巧： 如果您启用了 Admin UI，您可以使用管理员凭据登录到此 UI，当您选择左侧边栏上的 `CONFIGURATION` 部分时，它将展开并允许您执行上述任何临时命令。

对 HTTP 服务的请求必须以使用 ad-hoc 命令定义的 API 密钥之一结束：`http://localhost:8080/rest/adhoc/sess-man@domain.com?api-key=test1`

备注： 如果您想允许在不使用任何密钥的情况下访问 REST API，这是可能的。为此，您需要添加一个 API 密钥，其 `API key` 字段值等于 `open_access`。

备注： 您还可以通过将 `'open-access' = true` 添加到 TDSL 配置文件中来完全禁用 api-keys，无论是在 `http` bean 还是该 bean 的任何模块中，例如 `rest`, ‘admin 等

1.4.2 DNS 网络服务模块

对于基于网页的 XMPP 客户端，无法执行 DNS SRV 请求来查找特定域的 XMPP 服务器托管地址。为了解决这个问题，创建了 DNS 网络服务模块。

它处理传入的 HTTP GET 请求并使用传递的 `domain` 和 `callback` HTTP 参数执行 DNS 请求，如 XEP-0156: 发现替代 XMPP 连接方法 中所述。结果以 JSON 格式返回，以便于基于 Web 的 XMPP 客户端处理。

默认情况下，它部署在 `dns-webservice`

参数

domain
: 用于查找 XMPP SRV 客户端记录的域名。

回调
: 由于安全原因，基于 Web 的客户端可能由于跨域 AJAX 请求而无法访问某些 DNS Web 服务。传递可选的 `callback` 参数设置 JSONP 请求的回调名称，并以 JSONP 格式产生正确的响应。

发现连接到 XMPP 服务器的方法

使用 `host-meta`

您应该访问位于 `/dns-webservice/.well-known/host-meta` 的可用端点。

要使其遵循规范，您应该配置从 http 服务器的根路径到上述路径的重定向。例如，使用 nginx：

```
location  /.well-known/ {
    proxy_pass http://localhost:8080/dns-webservice/.well-known/;
    proxy_set_header Host $host;
}
```

查询特定域

如果我们想知道 sure.im 的连接选项，我们应该发送 HTTP GET 请求到 http://our-xmpp-server:8080/dns-webservice/?domain=sure.im&version=2 。我们将收到以下回复：

```
{
  domain: 'sure.im',
  c2s: [
    {
      host: 'tigase.me',
      ip: ['198.100.157.101','198.100.157.103','198.100.153.203'],
      port: 5222,
      priority: 5
    }
  ],
  bosh: [
    {url:'http://blue.sure.im:5280/bosh'},
    {url:'http://green.sure.im:5280/bosh'},
    {url:'http://orange.sure.im:5280/bosh'}
  ],
  websocket: [
    {url:'ws://blue.sure.im:5290/'},
    {url:'ws://green.sure.im:5290/'},
    {url:'ws://orange.sure.im:5290/'}
  ]
}
```

正如您在这里看到的，我们有托管 sure.im 域的 XMPP 服务器的名称和 IP 地址，以及使用 BOSH 或 WebSocket 建立连接的 URI 列表。

该模块默认激活。但是，如果您在测试环境中操作，您可能没有为您正在使用的域设置 SRV 和 A 记录，您可能希望在 config.tdsl 文件中使用以下行禁用此功能：

```
rest {
    'dns-webservice' (active: false) {}
}
```

1.4.3 启用密码重置机制

如果用户忘记了 XMPP 帐户的密码，可以为用户提供更改密码的机制。要做到这一点，你需要在你的类路径中有 `tigase-extras.jar` （它是 `-dist-max` 分发包的一部分），启用 `mailer` 和 `account-email-password-resetter`。

示例配置。

```
account-email-password-resetter () {}
mailer (class: tigase.extras.mailer.Mailer) {
    'mailer-from-address' = 'email-address@to-send-emails-from'
    'mailer-smtp-host' = 'smtp.email.server.com'
    'mailer-smtp-password' = 'password-for-email-account'
    'mailer-smtp-port' = '587' # Email server SMTP port
    'mailer-smtp-username' = 'username-for-email-account'
}
```

备注： 您需要用正确的配置参数替换示例配置参数。

使用此配置并在 URL http://localhost:8080/rest/user/resetPassword 重新启动 Tigase XMPP 服务器后，将提供可用于密码重置的 Web 表单。

备注： 只有当用户在帐户注册期间提供了真实的电子邮件地址并且用户仍然记得并可以访问在注册期间使用的电子邮件地址时，此机制才会起作用。

1.5 管理界面指南

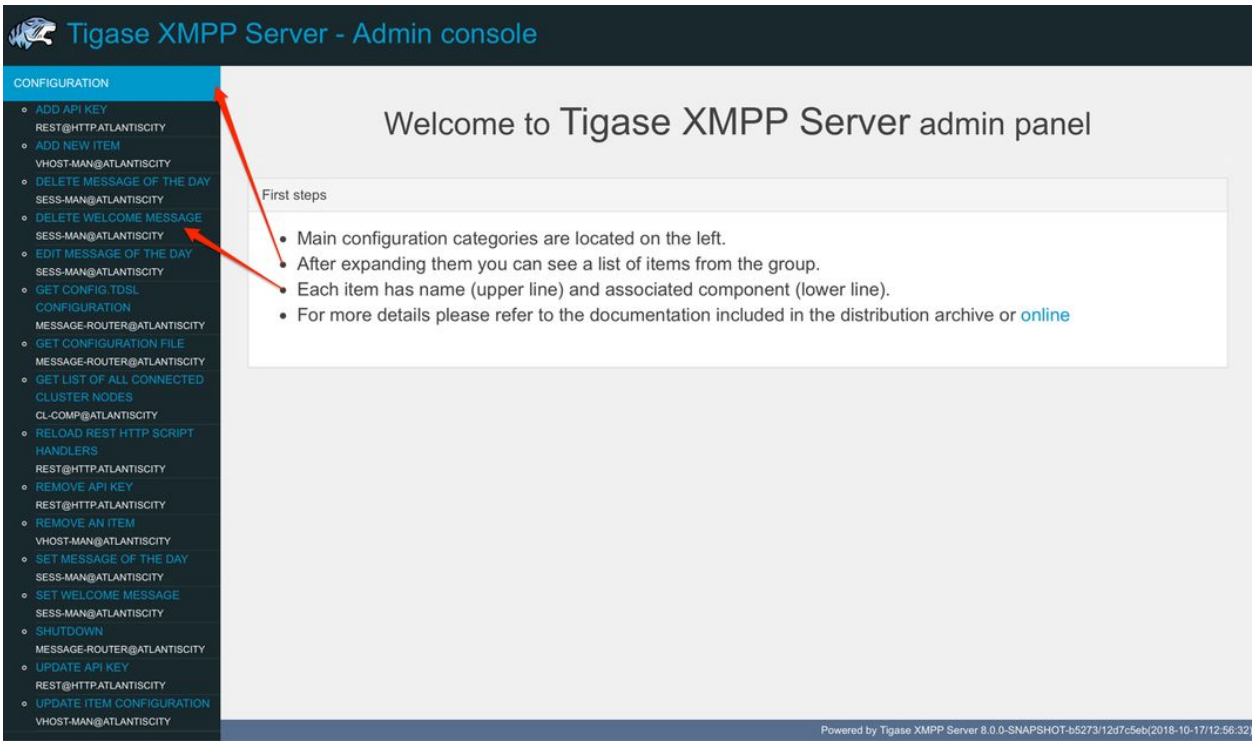
管理员用户界面是一个基于 HTTP 的界面，它向服务器发送 REST 命令以更新配置、更改设置和检索统计信息。

1.5.1 关于 REST 的说明

REST 代表 REpresentational State Transfer，这是一种无状态通信方法，在我们的例子中，它使用 HTTP GET、PUT、POST 和 DELETE 命令将命令传递到 Tigase 服务器内的资源。

1.5.2 用户界面的总体概述

导航到 Admin WebUI 后，您将看到有关导航的基本信息。面板本身由两个主要部分组成：* left 导航菜单，将所有配置项分组到类别中；* central，主配置页面，显示所选项目的配置选项。



每个配置项都有名称（上行）和关联的组件（下行），因为某些功能可以在不同组件的上下文中执行（例如，可以为 VirtualHost Manager 或 ExternalConnection Manager 执行 `Update Item Configuration`）

1.5.3 配置

允许您配置一些服务器设置，例如每日消息、欢迎消息或初始化集群节点的关闭。

1.5.4 示例脚本

这是一个脚本示例列表，可以为每个组件运行并执行一些琐碎的功能。它们本身可能没有特定的价值，但在编写自定义脚本时可以用作参考。获取可用命令列表是一个脚本，它存在于服务器上的每个活动组件，并且正如其标题所暗示的那样，将为该组件提供所有命令的列表。最后，两个脚本来自于本指南的脚本部分。一般来说，本节没有太多需要看的内容。

1.5.5 通知

本节有一个简单的命令：能够向所有登录用户发送大量消息。可以从此部分发送三种类型的消息：- **normal** 消息将在大多数客户端中显示为弹出窗口。- **headline** 某些客户端会将标题消息插入到 MUC 或用户之间的聊天中，否则它会像普通消息一样创建弹出窗口。- **chat** 聊天消息将打开与用户的聊天对话框。

1.5.6 其他

本节包含大量影响服务器功能的选项和设置列表。

为用户激活日志跟踪器

这允许您设置日志文件来跟踪特定用户。设置您要记录的用户的裸机或完整 JID，以及您希望写入日志的文件的名称。这些文件将被写入根 Tigase 目录，除非您提供类似日志/文件名的目录。日志文件将以.0 扩展名命名，并且将命名为.1、.2、.3 等，因为每个文件默认达到 10MB。filename.0 将始终是最新的。发出命令后日志记录将开始，并在服务器重新启动后停止。

添加 SSL 证书

在这里，您可以将 PEM 文件中的 SSL 证书添加到特定的虚拟主机。尽管 Tigase 可以生成自己的自签名证书，但这将覆盖任何默认证书。证书不能包含密码，也不能加密。确保内容包含证书和私钥数据。您还可以选择将证书保存到磁盘，使更改永久生效。

添加监听脚本

本节允许您为 eventbus 组件创建自定义函数。如果满足设置的条件，这些脚本可以让服务器执行某些操作。您可以使用 Groovy 或 EMCAscript 编写脚本。有关详细信息，请参阅 *eventbus* 部分。

添加监控任务

您可以在此处为 Groovy 或 ECMAScript 编写脚本以添加到监视任务。这只会将脚本添加到可用脚本中，但是，您需要从另一个提示符运行它。请注意，这些脚本可能仅适用于监视器组件。

添加监控定时器任务

本部分允许您在 Groovy 中添加监控脚本，同时使用延迟设置来延迟脚本的启动。

添加新项目-ext

根据您是否加载了任何外部组件，这可能会显示。这允许您向 Tigase 的运行实例添加额外的外部组件。

添加新项目-Vhost

这允许您将新的虚拟主机添加到 XMPP 服务器。字段细分如下：

- 域名：新虚拟主机的完整域名。Tigase 不会向该域添加任何内容，因此如果它是 example.com 的子域，则需要输入 sub.domain.com。
- 已启用：域是打开还是关闭。
- 启用匿名：允许匿名登录。
- 带内注册：是否允许用户在登录时注册帐户。
- 需要 TLS：需要登录到虚拟主机才能在打开流之前进行 TLS 握手。
- S2S 密码：服务器生成的代码，用于区分服务器之间的流量，通常不需要输入您自己的代码，但如果您需要进入低级代码，您可以输入。
- 域过滤策略：设置此域的过滤策略，有关规则的描述，请参阅 本节。
- 域过滤域：限制或控制跨域流量的特定设置。
- 最大用户数：允许注册到服务器的最大用户数。
- 允许的 C2S、BOSH、Websocket 端口：此虚拟主机将检查所有这些服务的端口的逗号分隔列表。
- 呈现转发地址：转发存在信息的特定地址。如果您希望使用单个域进行存在处理，这可能会很方便。
- 消息转发地址：将发送所有消息的特定地址。如果您有一个处理 AMP 或消息存储的服务器并希望将负载保持在那里，这可能对您有用。
- 其他参数：您可能希望传递给服务器的其他设置，将其视为命令后的选项部分。
- 所有者：虚拟主机的所有者，也将被视为管理员。
- 管理员：将被视为虚拟主机管理员的 JID 的逗号分隔列表。
- XEP-0136 启用消息存档：是否打开或关闭此功能。
- XEP-0136 所需的存储方法：如果 XEP-0136 已打开，您可以限制保存的消息部分。这对于任何归档都是必需的，如果为 null，则可以存储消息的任何部分。
- 需要客户端证书：客户端是否应提交证书才能登录。
- 客户端证书 CA：客户端证书的证书颁发机构。
- XEP-0136 保留期限：将设置消息存档的天数（整数）。
- 受信任的 JID：将添加到受信任列表的 JID 的逗号分隔列表，这些 JIDS 可以在不需要安全登录的情况下执行命令、编辑设置或其他安全工作。

- XEP-0136 保留类型：设置保留期将使用的数据类型。可以是用户定义（自定义数字类型）、无限制或天数。
- XEP-0136 - 存储 MUC 消息：是否存储 MUC 消息以进行归档。默认为 user，允许用户单独进行设置，否则 true/false 将覆盖。
- 启用 see-other-host 重定向：在具有多个集群的服务器中，如果一个集群出现故障，此功能将有助于自动重新填充集群列表，但是如果未选中此选项，该列表将不会更改，并且可能会尝试将流量发送到故障服务器。
- XEP-0136 默认存储方法：将存储在存档中的默认消息部分。

更改用户域间通信权限

在这里，您可以限制用户能够在特定域上进行通信，这类似于使用相同规则集的域过滤策略。有关更多详细信息，请参阅基于域的数据包过滤 部分了解规则详细信息和细节。请注意，可以同时对多个 JID 进行更改。

连接时间

列出从客户端到服务器的最长和平均连接时间。

创建节点

本节允许您为 pubsub 组件创建一个新节点。以下是字段的细分：

- 要创建的节点：这是将要创建的节点的名称。
- Owner JID：将被视为节点所有者的用户 JID。
- pubsub#node type：设置新节点的节点类型。选项包括：
  - 可以发布和也可以被发布的 **leaf** 节点。
  - **collection** 其他节点的集合。
- 节点的友好名称：允许使用空格和其他字符来帮助将其与其他节点区分开来。
- 是否发送带有事件通知的有效负载：正如它所说，是否发布事件。
- 配置更改时通知订阅者：默认为 false
- 将项目保留到存储：是否将项目物理存储在节点中。
- 要保留的最大项目数：限制节点存档中保留的项目数量。
- 节点所属的集合：如果节点要在集合中，请在此处放置该节点名称。
- 指定订阅者模型：选择将用于此节点的订阅者模型类型。选项包括：
  - **authorize** - 需要节点所有者批准所有订阅，然后才能将项目发布给用户。此外，只有订阅者可以检索项目。

  - **open** - 所有用户都可以从节点订阅和检索项目。
  - **presence** - 通常用于即时消息环境。提供一个系统，在该系统下，通过一个或两个订阅订阅所有者 JID 存在的用户可以从节点订阅和检索项目。
  - **roster** - 这也用于即时消息环境中，订阅了所有者存在并且被名册放置在特定允许的组中的用户能够订阅节点并从中检索项目。
  - **whitelist** - 仅允许明确允许的 JID 从节点订阅和检索项目，此列表由所有者/管理员设置。
- 指定发布者模型：选择将用于此节点的发布者模型类型。选项包括：
  - **open** - 任何用户都可以发布到该节点。
  - **publishers** - 只有列为发布者的用户才能发布。
  - **subscribers** - 只有订阅者可以发布到该节点。
- 何时发送最后一个发布的项目：这允许您决定是否以及何时可以将最后一个发布到节点的项目发送给新订阅的用户。
  - **never** - 不要发送最后发布的项目。
  - **on_sub** - 当用户订阅节点时发送最后发布的项目。
  - **on_sub_and_presence** - 订阅后将最后发布的项目发送给用户，并且用户可用。
- 允许访问此节点的域：用户可以访问此节点的域的逗号分隔列表。默认为空白，没有域限制。
- 是否仅向可用用户发送项目：如果选择此项，则仅将项目发布给具有可用状态的用户。
- 订阅者下线时是否订阅过期：这将使该节点的所有订阅对单个会话有效，并且需要在重新连接时重新订阅。
- 可应用于有效负载以生成适当的消息正文元素的 XSL 转换：由于您需要格式正确的 <body> 元素，因此您可以在此处添加 XSL 转换以解决任何有效负载或此处要正确格式化的额外元素。
- XSL 转换的 URL，可应用于有效负载以生成适当的消息正文元素：这将是 XSL 转换的 URL，例如 http://www.w3.org/1999/XSL/Transform。
- 允许订阅的花名册组：用户可以订阅的组列表。如果此项为空白，则不会施加用户限制。
- 当所有者改变他们的订阅或附属状态时通知订阅者：这将使节点在所有者改变附属或订阅状态的情况下发送消息。
- 允许获取每个订阅者的订阅者列表：允许订阅者生成节点的其他订阅者列表。
- 是否按创建日期或更新时间对集合项进行排序：选项包括
  - **byCreationDate** - 项目将按创建日期排序，即项目的制作时间。
  - **byUpdateTime** - 项目将按上次更新时间排序，即项目上次编辑/发布/等等的时间。

DNS 查询

一个基本的 DNS 查询表单。

默认配置 - Pubsub

在这里，您可以为任何新的 pubsub 节点进行默认配置。这些更改将针对所有未来的节点进行，但不会影响当前活动的节点。

默认房间配置

此页面允许管理员为可能在服务器上创建的任何新 MUC 房间设置默认配置。

删除监控任务

这将从可用监控脚本列表中删除监控任务。此操作不是永久性的，因为它将在服务器重新启动时恢复为初始设置。

删除节点

提供从服务器中删除节点的空间。必须是节点的全名，一次只能删除一个节点。

删除所有节点

此页面允许登录的管理员从关联的虚拟主机中删除所有节点。此更改是不可逆的，请务必在提交命令之前阅读并选中该框。

修复用户名册

您可以从此提示来修复用户名册。填写用户的裸 JID 以及您希望从名册中添加或删除的名称。这不会编辑用户名册，而是将客户名册与数据库进行比较并修复它们之间的任何错误。

修复 Tigase 集群上的用户名册

这与修复用户名册的作用相同，但可以应用于可能未登录到本地虚拟主机但登录到集群服务器的用户。

获取用户名册

正如标题所示，这将获取用户名册并将其显示在屏幕上。您可以使用裸 JID 或完整 JID 来获取特定名册。

获取任何文件

这使您能够查看 tigase 目录中任何文件的内容。默认情况下，您位于根目录中，如果您希望进入目录，请使用以下格式：logs/tigase.log.0

获取配置文件

如果您不想输入配置文件的位置，可以使用此提示调出 tigase.conf 或 config.tdsl 的内容。

获取 config.tdsl 文件

将输出当前的 config.tdsl 文件，这包括在当前服务器会话期间所做的任何修改。

获取可用命令列表

这可能会针对不同的组件多次列出，但这将按照本节的建议并列出该特定组件的可用命令。

负载测试

在这里，您可以在任何节点上使用 pubsub 组件运行测试，以测试节点的功能和正确设置。

加载错误

将显示服务器在加载和运行过程中遇到的任何错误。如果您需要解决任何问题，可能会很有用。

新的命令脚本

此处允许您创建将在相关组件中工作的新命令脚本。请注意，在超链接标题下，有 muc.server.org 或 pubsub.server.org 的列表，使用它们来确定新命令将在哪里运行。

OAuth 凭据

这允许为服务器设置新的自定义 OAuth 凭据，并且您还可以要求用户在登录时使用 OAuth 令牌。这是您登录的特定主机的设置。如果您登录到 xmpp1.domain.com，它不会影响 xmpp2.domain.com 的设置。

预绑定 BOSH 用户会话

这允许在用户登录之前将 JID 与 BOSH 会话配对，如果您有定期通过 BOSH 登录的用户或定期连接的 Web 客户端，则可以减少 CPU 使用。您还可以指定 HOLD 和 WAIT 整数来影响 BOSH 如何使用相关联的 JID 进行操作。

将项目发布到节点

此窗口不仅允许您测试，而且允许您将项目发布到指定节点。必须填写所有字段以避免服务器丢弃格式不正确的节。

读取所有节点

这将在内存中加载一棵 pubsub 节点树，它不会输出任何内容，因为它主要供开发人员使用。

重建数据库

这将强制 Tigase 为 pubsub 组件重建数据库，这对于取消订阅后继续收到推送事件的 pubsub 订阅者可能很有用。

重新加载组件存储库

这将重新加载服务器正在运行的任何虚拟主机。如果在运行时断开连接或损坏，这可能很有用。

移除一个项目

这将从服务器中删除正在运行的虚拟主机，您将看到一个可供选择的列表。

删除命令脚本

像新的命令脚本一样，查看子标题以确定要从哪个组件中删除脚本。在那里，选择您希望从服务器中删除的命令。如果选择从磁盘中删除，则更改将是永久性的。否则，该命令将被删除，直到下一次服务器重新启动。

删除侦听器脚本

从列表中选择要删除的侦听器脚本。这只会影响添加到 eventbus 组件的自定义侦听器脚本。

移除房间

这提供了从 MUC 组件中删除房间的字段。您可以建议一个替代房间，一旦当前房间被移除，它将把居住者转移到替代房间。

检索项目

在这里，您可以从 PubSub 节点检索项目，这模拟了从 pubsub 组件获取 IQ 节。- 服务名称 - 发布订阅组件的地址。- 节点名称 - 从中检索项目的项目节点。- 项目 ID - 您要检索的项目的项目 ID。- 项目自 - UTC 时间戳开始搜索：YYYY-MM-DDTHH:MM:SSZ

S2S 不良状态连接

这将列出与其他服务器的任何连接，这些连接被认为是坏的或陈旧的。这将很少进行填写，因为 Tigase 会自动调整关闭的集群服务器。如果连接仍然不好，建议在下一个空间中重置这些连接。

S2S 重置不良状态连接

这将重置与其他服务器的连接，这些服务器被认为是坏的并显示在 S2S 坏状态连接页面中。

S2S 获取 CID 连接状态

仅供内部开发人员使用。

订阅节点

这为管理员提供了手动让 JID 订阅特定节点的空间。

退订节点

在这里，您可以取消订阅特定节点的用户。用户可以是一个逗号分隔的列表。

更新项目配置

通常，您只会看到 vhost-man 的一项，但一些附加组件（即 ext）也可能提供它们。它们每个都有自己的部分，但提供了过多的服务器选项。对服务器的更改是实时完成的，并且可能不是永久性的。

vhost-man

您将看到 Tigase 当前托管的域列表，您将能够使用此功能一次更改一个域的设置。选择域后，您将能够设置或更改以下设置：

- 域名：新虚拟主机的完整域名。Tigase 不会向该域添加任何内容，因此如果它是 example.com 的子域，则需要输入 sub.domain.com。
- 已启用：域是打开还是关闭。
- 启用匿名：允许匿名登录。
- 带内注册：是否允许用户在登录时注册帐户。
- 需要 TLS：需要登录到虚拟主机才能在打开流之前进行 TLS 握手。
- S2S 密码：服务器生成的代码，用于区分服务器之间的流量，通常不需要输入您自己的代码，但如果您需要进入低级代码，您可以输入。
- 域过滤策略：设置此域的过滤策略，有关规则的描述，请参阅本节。
- 域过滤域：限制或控制跨域流量的特定设置。
- 最大用户数：允许注册到服务器的最大用户数。
- 允许的 C2S、BOSH、Websocket 端口：此虚拟主机将检查所有这些服务的端口的逗号分隔列表。
- 呈现转发地址：转发存在信息的特定地址。如果您希望使用单个域进行存在处理，这可能会很方便。
- 消息转发地址：将发送所有消息的特定地址。如果您有一个处理 AMP 或消息存储的服务器并希望将负载保持在那里，这可能对您有用。
- 其他参数：您可能希望传递给服务器的其他设置，将其视为命令后的选项部分。
- 所有者：虚拟主机的所有者，也将被视为管理员。
- 管理员：将被视为虚拟主机管理员的 JID 的逗号分隔列表。
- XEP-0136 启用消息存档：是否打开或关闭此功能。
- XEP-0136 所需的存储方法：如果 XEP-0136 已打开，您可以限制保存的消息部分。这对于任何归档都是必需的，如果为 null，则可以存储消息的任何部分。
- 需要客户端证书：客户端是否应提交证书才能登录。
- 客户端证书 CA：客户端证书颁发机构。
- XEP-0136 保留期：将设置消息存档天数（整数）。

- 受信任的 JID：将添加到受信任列表的 JID 的逗号分隔列表，这些 JIDS 可以在不需要安全登录的情况下执行命令、编辑设置或其他安全工作。
- XEP-0136 保留类型：设置保留期将使用的数据类型。可以是用户定义（自定义数字类型）、无限制或天数。
- XEP-0136 - 存储 MUC 消息：是否存储 MUC 消息以进行归档。默认为 user，允许用户单独进行设置，否则 true/false 将覆盖。
- 启用 see-other-host 重定向：在具有多个集群的服务器中，如果一个集群出现故障，此功能将有助于自动重新填充集群列表，但是如果未选中此选项，该列表将不会更改，并且可能会尝试将流量发送到故障服务器。
- XEP-0136 默认存储方法：将存储在存档中的默认消息部分。

更新用户名册条目

此部分允许管理员编辑单个用户名册，尽管它提供了类似的功能来修复用户名册，但这是为精确编辑用户名册而设计的。

- 名册所有者 JID：您要编辑的用户名册的 BareJID。
- 要操作的 JID：您要添加/删除/更改的特定 BareJID。
- 逗号分隔组：您也希望添加的 JID 的组。
- 操作类型：将执行什么功能?
  - **Add** - 将要操作的 JID 添加到所有者 JID 的名册和组中。
  - **Remove** - 从所有者 JID 的名册和组中删除要操作的 JID。
- 订阅类型：将发送到服务器的订阅节类型，随后将在两个用户之间使用。
  - **None** - 如果所有者或要被操纵的用户都不希望接收存在信息，请选择此选项。
  - **From** - 名册所有者不会从 JID 接收到存在信息来进行操作，但情况正好相反。
  - **To** - 要操作的 JID 不会从名册所有者那里接收到存在信息，但情况正好相反。
  - **Both** - 两个 JID 都将收到关于彼此的存在信息。

更新用户名册条目扩展版

本节是上一节的扩展版本，所有已指定的字段与以下添加相同：

- 名册所有者名称：如果您想更改/创建一个友好名称或昵称。**不是必需**
- 所有者组的逗号分隔：用户想要加入/离开的组。**不是必需**
- 名册项目 JID：需要编辑的特定的 JID。
- 名册项目名称：将更改/创建的友好名称或昵称。**不是必需**

- 逗号分隔的项目组列表：名册项目 JID 将被添加到/删除的组或组列表。
- 行动：
  - **Add/update item** - 将在名册所有者的名册中添加或更新项目 JID。
  - **Remove item** - 将从名册所有者的名册中删除项目 JID
  - **Add/update both rosters** - 将在名册所有者和名册项目的名册中添加或更新项目。
  - **Remove from both rosters** - 将从名册所有者和名册项目的名册中删除该项目。

1.5.7 脚本

此部分将使管理员能够为特定组件自定义编写或输入他们自己的脚本。每个活动组件都将有一个新的和删除命令脚本的条目，并且为该组件编写的脚本将放在那里。

新的命令脚本

- Description：脚本的友好名称，将是左侧菜单中链接的标题。
- Command ID：Tigase 在引用此脚本时将使用的内部命令。
- Group：脚本的组，其可以是左侧的任何标题（配置、示例脚本、通知、其他等）或您自己的。如果不存在组，将创建一个新组。
- Language: 编写脚本的语言。目前 Tigase 支持 Groovy 和 EMCAScript。
- Script text: 脚本的全文。
- Save to disk：保存到磁盘的脚本将永久存储在服务器的目录 /scripts/admin/[Component]/commandID.js **注意**未保存到磁盘的脚本将无法在服务器重新启动后继续存在。

删除命令脚本

与新命令脚本一样，每个组件都有一个条目。此页面将提供一个地方来删除所选组件的命令。您将获得与该组件关联的脚本列表。您还可以从磁盘中删除，这将从服务器所在的硬盘驱动器中永久删除脚本。如果未选中此项，则脚本将在下次重新启动之前不可用。

1.5.8 统计数据

这部分对于测试统计脚本和组件更有用，因为它们中的许多会产生非常少量的信息，但是这些可能会被其他组件或脚本收集以更好地显示信息。

获取用户统计信息

提供用户统计信息的脚本输出，包括正在使用的活动会话数、使用的数据包数量、特定连接及其数据包使用情况和位置。所有资源都将返回个人统计信息以及 IP 地址。

获取活跃用户列表

提供服务器内选定域下的活动用户列表。活动用户被认为是当前登录到 XMPP 服务器的用户。

获取空闲用户列表

提供服务器上空闲的用户列表。

获取在线用户列表

提供当前在线的用户列表。

获取活跃用户数

提供活动用户列表，即非空闲或离开的用户。

获取空闲用户数

提供多个空闲用户。

获取顶级活跃用户

将生成一个用户限制用户列表，这些用户在发送的数据包中被认为是最活跃的。

1.5.9 用户

添加用户

在这里，您可以将新用户添加到 vHosts 处理的任何域，用户会立即添加到数据库并能够登录。**注意：您不能在此部分中授予这些用户管理员身份**。

更改用户密码

这使您可以更改数据库中任何用户的密码。虽然更改会立即生效，但当前登录的用户在尝试再次登录之前不会知道密码已更改。

删除用户

这将从数据库中删除一个或多个用户（逗号分隔）。单击提交后，删除的用户将被踢出服务器。

结束用户会话

通过结束与服务器的会话来断开当前选定用户的连接。

获取用户信息

此部分允许管理员获取有关特定用户的信息，包括当前连接以及等待传递的离线和在线消息。

获取注册用户列表

这将显示所选域的所有注册用户，直到指定数量。

修改用户

允许您修改一些用户详细信息，包括电子邮件以及是否为活动用户。

1.6 Tigase 网络客户端

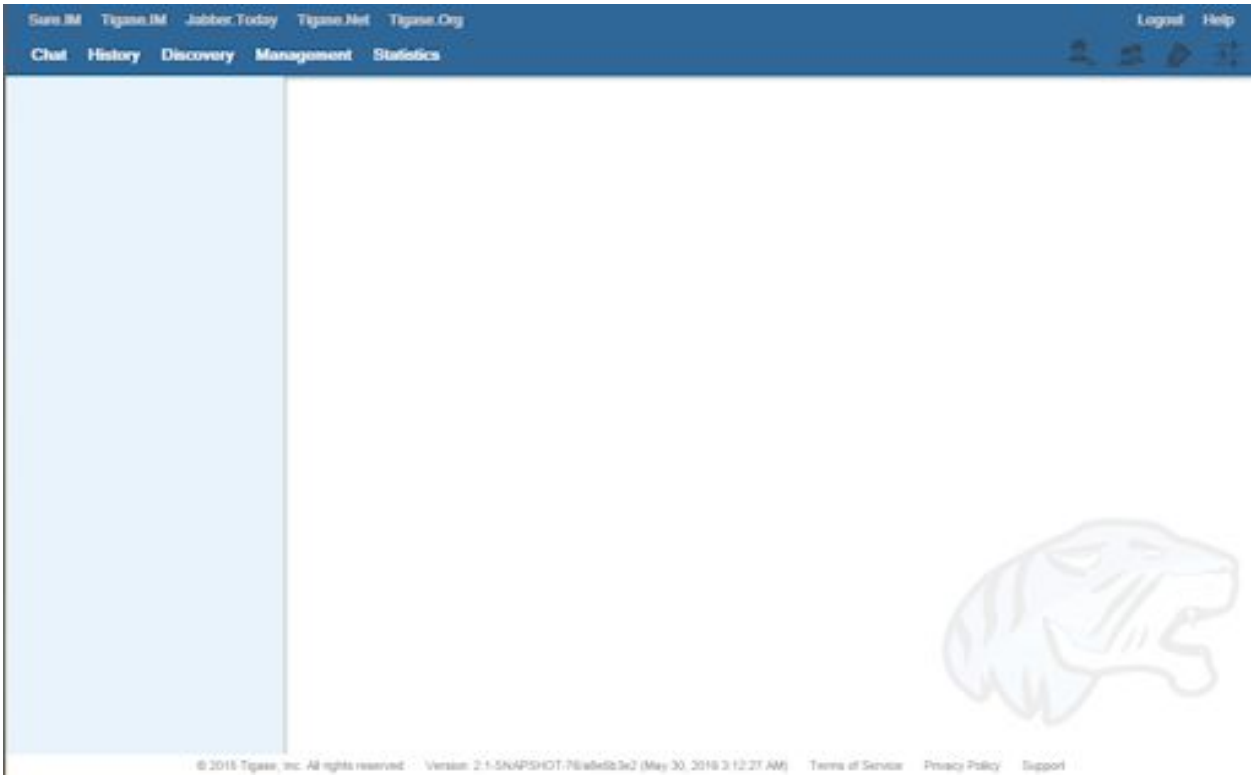
Tigase 现在在 HTTP 接口中内置了一个功能齐全的 XMPP 客户端。您对 XMPP 客户端的所有期望现在都可以在浏览器窗口中轻松完成，无需安装软件！

默认情况下，Web 客户端在服务器 v7.2.0 及更高版本上处于活动状态且可用。

要访问客户端，请将浏览器指向以下地址：xmpp.your-server.net:8080/ui/

它会要求您登录，任何在服务器上注册的用户的裸 JID 都可以使用。**注意：使用您的裸 JID 登录**

一旦成功登录后，您将看到以下屏幕。



这些命令分为此处显示的类别。在这些部分中所做的所有更改都是即时的，并且应该与您使用外部 XMPP 客户端（如 Psi）相同。

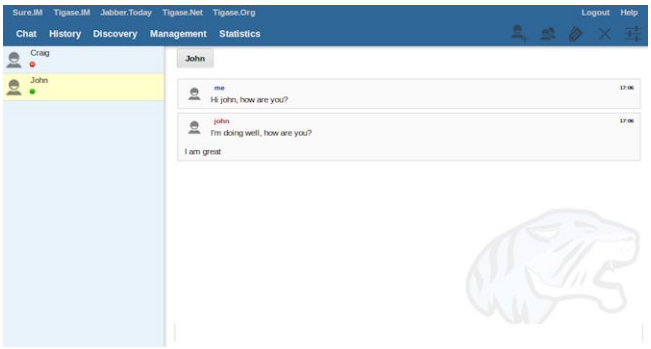
注意 BOSH 客户端会自动将所有请求转换为服务器名称。在极少数情况下，浏览器可能无法解决此问题，您将无法登录。如果发生这种情况，您可以使用 config.tdsl 中的以下行禁用该功能：

```
bosh {
    'send-node-hostname' = false
}
```

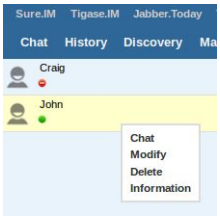
在登录屏幕中使用高级标签时，您可能必须专门指定 bosh URL。

1.6.1 聊天

这是默认窗口，也是您在 XMPP 中与该服务器聊天的主界面。**注意：您只能与登录到当前服务器或连接的集群的用户交流**您名册中的用户将在左侧面板上，右侧所有活跃的讨论和 MUC，以及当前选择的聊天将可用。



登录并在您的花名册上的用户将显示在左侧。双击将弹出一个与用户的新聊天窗口。您可以右键单击它们以显示带有以下内容的子菜单；



- **Chat** 复制双击并打开一个新的聊天窗口。
- **Modify** 弹出一个对话框，允许您更改联系人的 JID、昵称和组。
- **Delete** 从您的名册中删除该用户。这还将删除所选用户的订阅授权，以便有效地将您从他们的名册中删除。**注意：这不会阻止来自您的 JID 的用户数据包**
- **Info** 调出用户信息（这是所选用户的 disco#info 命令）

右上角有几个具有特定功能的图标，它们是；

 将新用户添加到您的名册中。

 创建一个新的多用户聊天室。

 允许您编辑您的用户信息，例如图片和昵称。

 关闭活动聊天窗口。

 提供更改密码或发布对用户信息的更改的地方。**注意：您只能更改常规字段**

1.6.2 发现

这是您的服务发现面板，它按侧边栏中的组件进行细分。列出了每个组件名称及其关联的 JID，以帮助您找到所需的内容。大多数组件都为您提供了执行命令的选项，但有少数例外允许浏览和加入 MUC。

Browse 允许您深入挖掘某些组件；例如列出 MUC 组件中可用的聊天室。在页面顶部，将显示您所在组件的特定 JID。这是一个文本字段，可以进行编辑以反映要导航的组件的 JID（或只是服务器名称）。

Join to Room 将加入您选择的 MUC 房间。或者，在选择 MUC 组件时选择加入房间，您可以加入并开始一个新的 MUC 房间。

Execute Command 提供命令和选项的层次结构以查看和编辑设置、运行命令和脚本、查看文件内容以及查看统计信息。由于每个组件都可以具有独特的结构，因此最好探索每个组件以查看可用的选项。

1.6.3 管理

这是 XMPP 服务器设置和管理的高级窗口。

配置

在这里您可以管理一些服务器设置。

通知

本节有一个简单的命令：能够向所有登录用户发送大量消息。您可以选择将消息类型更改为标题或普通，这将在大多数 XMPP 客户端中显示为弹出窗口。聊天消息将打开与用户的聊天对话框。

其他

本节包含大量影响服务器功能的选项和设置列表。

为用户激活日志跟踪器

这允许您设置日志文件来跟踪特定用户。设置您要记录的用户的裸机或完整 JID，以及您希望写入日志的文件的名称。除非您提供类似日志/文件名的目录，否则这些文件将被写入根 Tigase 目录。日志文件将以.0 扩展名命名，默认情况下每个文件达到 10MB 时将命名为.1、.2、.3 等。filename.0 将始终是最新的。一旦服务器重新启动，日志记录将停止。

添加 SSL 证书

在这里，您可以将 PEM 文件中的 SSL 证书添加到特定的虚拟主机。尽管 Tigase 可以生成自己的自签名证书，但这将覆盖那些默认证书。

添加监控任务

您可以在此处为 Groovy 或 ECMAScript 编写脚本以添加到监视任务。这只会将脚本添加到可用脚本中，但是，您需要从另一个提示符运行它。

添加监控定时器任务

本部分允许您在 Groovy 中添加监控脚本，同时使用延迟设置来延迟脚本的启动。

添加新项目 - ext

提供一种将外部组件添加到服务器的方法。默认情况下，您被视为所有者，并且自动填充 Tigase 负载均衡器。

添加新项目 - Vhost

这允许您将新的虚拟主机添加到 XMPP 服务器。字段细分如下：

- 域名：新虚拟主机的完整域名。Tigase 不会向该域添加任何内容，因此如果它是 example.com 的子域，则需要输入 sub.domain.com。
- 已启用：域是打开还是关闭。
- 启用匿名：允许匿名登录。
- 带内注册：是否允许用户在登录时注册帐户。
- 需要 TLS：需要登录到虚拟主机才能在打开流之前进行 TLS 握手。
- S2S 密码：服务器生成的代码，用于区分服务器之间的流量，通常不需要输入您自己的代码，但如果您需要进入低级代码，您可以输入。
- 域过滤策略：设置此域的过滤策略，有关规则的描述，请参阅本节。
- 域过滤域：限制或控制跨域流量的特定设置。
- 最大用户数：允许注册到服务器的最大用户数。
- 允许的 C2S、BOSH、Websocket 端口：此虚拟主机将检查所有这些服务的端口的逗号分隔列表。
- 呈现转发地址：转发存在信息的特定地址。如果您希望使用单个域进行存在处理，这可能会很方便。
- 消息转发地址：将发送所有消息的特定地址。如果您有一个处理 AMP 或消息存储的服务器并希望将负载保持在那里，这可能对您有用。
- 其他参数：您可能希望传递给服务器的其他设置，将其视为命令后的选项部分。
- 所有者：虚拟主机的所有者，也将被视为管理员。
- 管理员：将被视为虚拟主机管理员的 JID 的逗号分隔列表。
- XEP-0136 启用消息存档：是否打开或关闭此功能。

- XEP-0136 所需的存储方法：如果 XEP-0136 已打开，您可以限制保存的消息部分。这对于任何归档都是必需的，如果为 null，则可以存储消息的任何部分。
- 需要客户端证书：客户端是否应提交证书才能登录。
- 客户端证书 CA：客户端证书的证书颁发机构。
- XEP-0136 保留期限：将设置消息存档的天数（整数）。
- 受信任的 JID：将添加到受信任列表的 JID 的逗号分隔列表，这些 JIDS 可以在不需要安全登录的情况下执行命令、编辑设置或其他安全工作。
- XEP-0136 保留类型：设置保留期将使用的数据类型。可以是用户定义（自定义数字类型）、无限制或天数。
- XEP-0136 - 存储 MUC 消息：是否存储 MUC 消息以进行归档。默认为 user，允许用户单独进行此设置，否则 tue/false 将覆盖。
- 启用 see-other-host 重定向：在具有多个集群的服务器中，如果一个集群出现故障，此功能将有助于自动重新填充集群列表，但是如果未选中此选项，该列表将不会更改，并且可能会尝试将流量发送到故障服务器。
- XEP-0136 默认存储方法：将存储在存档中的默认消息部分。

更改用户域间通信权限

您可以限制用户只能向某些虚拟主机发送和接收数据包。如果您想将用户锁定到特定域，或者阻止他们从统计组件获取信息，这可能会很有帮助。

连接时间

列出从客户端到服务器的最长和平均连接时间。

DNS 查询

一个基本的 DNS 查询表单。

默认配置 - Pubsub

此部分使您能够对所有未来节点的默认 pubsub 节点配置进行更改。**注意：这些更改将在服务器重新启动时重置**。 - pubsub#node type：设置新节点的节点类型。选项包括：

- 可以发布和也可以被发布的 **leaf** 节点。
- **collection** 其他节点的集合。
  - 节点的友好名称：允许使用空格和其他字符来帮助将其与其他节点区分开来。

  - 是否发送带有事件通知的有效负载：正如它所说，是否发布事件。
  - 配置更改时通知订阅者：默认为 false
  - 将项目保留到存储：是否将项目物理存储在节点中。
  - 要保留的最大项目数：限制节点存档中保留的项目数量。
  - 节点所属的集合：如果节点要在集合中，请在此处放置该节点名称。
  - 指定订阅者模型：选择将用于此节点的订阅者模型类型。选项包括：

- **authorize** - 需要节点所有者批准所有订阅，然后才能将项目发布给用户。此外，只有订阅者可以检索项目。
- **open** - 所有用户都可以从节点订阅和检索项目。
- **presence** - 通常用于即时消息环境。提供一个系统，在该系统下，通过一个或两个订阅订阅所有者 JID 存在的用户可以从节点订阅和检索项目。
- **roster** - 这也用于即时消息环境中，订阅了所有者存在并且被名册放置在特定允许的组中的用户能够订阅节点并从中检索项目。
- **whitelist** - 仅允许明确允许的 JID 从节点订阅和检索项目，此列表由所有者/管理员设置。
  - 指定发布者模型：选择将用于此节点的发布者模型类型。选项包括：

- **open** - 任何用户都可以发布到该节点。
- **publishers** - 只有列为发布者的用户才能发布。
- **subscribers** - 只有订阅者可以发布到该节点。
  - 何时发送最后一个发布的项目：这允许您决定是否以及何时可以将最后一个发布到节点的项目发送给新订阅的用户。

- **never** - 不要发送最后发布的项目。
- **on_sub** - 当用户订阅节点时发送最后发布的项目。
- **on_sub_and_presence** - 订阅后将最后发布的项目发送给用户，并且用户可用。
  - 允许访问此节点的域：用户可以访问此节点的域的逗号分隔列表。如果留空，则没有域限制。
  - 是否仅向可用用户发送项目：如果选择此项，则仅将项目发布给具有可用状态的用户。
  - 订阅者下线时是否订阅过期：这将使该节点的所有订阅对单个会话有效，并且需要在重新连接时重新订阅。
  - 可应用于有效负载以生成适当的消息正文元素的 XSL 转换：由于您需要格式正确的 <body> 元素，因此您可以在此处添加 XSL 转换以解决任何有效负载或此处要正确格式化的额外元素。
  - XSL 转换的 URL，可应用于有效负载以生成适当的消息正文元素：这将是 XSL 转换的 URL，例如 http://www.w3.org/1999/XSL/Transform。
  - 允许订阅的花名册组：用户可以订阅的组列表。如果此项为空白，则不会施加用户限制。

  - 当所有者改变他们的订阅或附属状态时通知订阅者：这将使节点在所有者改变附属或订阅状态的情况下发送消息。
  - 允许获取每个订阅者的订阅者列表：允许订阅者生成节点的其他订阅者列表。
  - 是否按创建日期或更新时间对集合项进行排序：选项包括

- **byCreationDate** - 项目将按创建日期排序，即项目的制作时间。
- **byUpdateTime** - 项目将按上次更新时间排序，即项目上次编辑/发布/等等的时间。

默认房间配置

允许您为新的 MUC 房间设置默认配置。这将无法修改当前使用的房间和永久房间。

删除监控任务

这将从可用监控脚本列表中删除监控任务。此操作不是永久性的，因为它将在服务器重新启动时恢复为初始设置。

修复用户名册

您可以从此提示修复用户名册。填写用户的裸 JID 以及您希望从名册中添加或删除的名称。您可以使用此工具编辑用户名册，并且更改是永久性的。

修复 Tigase 集群上的用户名册

这与修复用户名册的作用相同，但可以应用于集群服务器中的用户。

获取用户名册

正如标题所示，这将获取用户名册并将其显示在屏幕上。您可以使用裸 JID 或完整 JID 来获取特定名册。

获取任何文件

这使您能够查看 tigase 目录中任何文件的内容。默认情况下，您位于根目录中，如果您希望进入目录，请使用以下格式：logs/tigase.log.0

获取配置文件

如果您不想输入配置文件的位置，可以使用此提示调出 tigase.conf 或 config.tdsl 的内容。

获取 config.tdsl 文件

将输出当前的 config.tdsl 文件，这包括在当前服务器会话期间所做的任何修改。

加载错误

将显示服务器在加载和运行过程中遇到的任何错误。如果您需要解决任何问题，可能会很有用。

新命令脚本 - 监视器

允许您在 Groovy 中编写命令脚本并将它们物理存储，以便它们可以在服务器重新启动后保存并随时运行。此处编写的脚本将只能在 Monitor 组件上运行。

新命令脚本 - MUC

允许您在 Groovy 中编写命令脚本并将它们物理存储，以便它们可以在服务器重新启动后保存并随时运行。此处编写的脚本将只能在 MUC 组件上运行。

OAUth 凭据

使用 OAuth 设置新凭据并启用或禁用带有签名表单的注册要求。

预绑定 BOSH 用户会话

允许管理员将 BOSH 会话与完整或裸 JID 预绑定（在连接时自动填充资源）。您还可以指定 HOLD 或 WAIT 参数。

重新加载组件存储库

这将显示您是否有任何外部组件，并在任何线程卡住的情况下重新加载它们。

脚本

本节提供所有活动组件的命令脚本列表。每个组件都有以下选项 - **新命令脚本**提供了一种为使用 EMCAScript 或 Groovy 编写的特定组件编写新命令脚本的方法。您确实可以选择将脚本保存到磁盘，这将使脚本在服务器中永久存在。- **删除命令脚本**允许您从存储库中删除选定的脚本。如果未选中从磁盘中删除，则脚本将在服务器重新启动之前不可用。如果是，它将从服务器中永久删除。

新创建的命令将列在左列的组列表下。

统计数据

这些统计信息可能更有用，因为脚本结果会产生少量数据，但您可能会发现它们在查找服务器负载或查找用户问题时很有用。

获取用户统计信息

提供用户统计信息的脚本输出，包括正在使用的活动会话数、使用的数据包数量、特定连接及其数据包使用情况和位置。所有资源都将返回个人统计信息以及 IP 地址。

获取活跃用户列表

提供服务器内选定域下的活动用户列表。活动用户被认为是当前登录到 XMPP 服务器的用户。

获取空闲用户列表

这将列出由 vhost 分隔的所有空闲用户。

获取在线用户列表

这将列出由他们连接到的虚拟主机分隔的用户。该列表将包括裸 JID 以及该 JID 的任何资源。

获取活跃用户数

这将显示当前活动用户的数量。

获取空闲用户数

此部分返回每个特定虚拟主机的活动用户数。

获取顶级活跃用户

这将按发送的数据包和在线时间列出活跃用户的最高数量。此列表仅由当前在线的用户和来自所有虚拟主机的用户构建。

用户

添加新用户

在这里，您可以将新用户添加到 vHosts 处理的任何域，用户会立即添加到数据库并能够登录。**注意：您不能在此部分中授予这些用户管理员身份**。

更改用户密码

允许管理员更改特定用户的密码，而无需知道所选裸 JID 的原始密码。当前登录的用户在尝试重新登录之前不会知道密码已更改。

删除用户

为管理员提供一个文本窗口，以输入他们希望从服务器中删除的用户的裸 JID。

获取用户信息

此部分允许管理员获取有关特定用户的信息，包括当前连接以及等待传递的离线和在线消息。

获取注册用户列表

提供要搜索的虚拟主机列表和要列出的最大用户数。运行后，脚本将显示来自所选虚拟主机的已注册用户裸 JID 列表。

修改用户

允许您修改一些用户详细信息，包括电子邮件以及是否为活动用户。

CHAPTER 2

HTTP 文件上传组件

Tigase 的 HTTP 文件上传组件是 XEP-0363 HTTP 文件上传 规范的实现。这允许 XMPP 客户端之间的文件传输，方法是将文件上传到 HTTP 服务器并仅将下载文件的链接发送给收件人。

该实现利用 Tigase XMPP Server 使用的 HTTP 服务器和 Tigase HTTP API 组件提供用于文件上传和下载的 Web 服务器。

默认情况下，该组件是 **禁用的**，需要在配置文件中启用才能使用。另一个要求是需要将正确的数据库模式应用于组件将使用的数据库。

2.1 启用 HTTP 文件上传组件

配置。

```
upload() {}
```

2.2 元数据存储库

运行该组件需要一个存储库，它可以在其中存储有关已分配插槽的信息。为此，使用了元数据存储库。可以为每个域指定 `FileUploadRepository` 的特定实现。

默认情况下，所有域的元数据都将存储在 `default` 存储库中。将根据定义为 `default` 的数据源类型选择其实现。

2.2.1 DummyFileUploadRepository

这是一个非常简单的存储库，不存储任何数据。因此，它可以非常快！但是，它无法删除旧上传并应用任何上传限制。

2.2.2 JDBCFileUploadRepository

此存储库实现将数据存储在用于存储过程和函数的数据库中。默认情况下，数据应存储在 `tig_hfu_slots` 表中，但可以通过修改存储过程或重新配置存储库实现来更改它，以使用与其所提供的不相同的存储过程和函数。

2.3 贮存

组件包含可插拔的存储机制，这意味着实现自定义存储规定相对容易。默认情况下，使用基于 `DirectoryStore` 的存储。

目前，以下存储提供程序可开箱即用。

2.3.1 目录存储

这种存储机制将文件放置在名称与 分配插槽 的 id 对应的子目录中。如果需要，可以将单个用户分配的所有插槽目录分组到包含该用户名的目录中。

默认情况下，如果在集群环境中使用此存储，则没有冗余。每个文件都将存储在单个集群节点上。

可用属性：

路径
: 包含目录的路径，在此处，将在本地计算机上创建包含文件的子目录。（**默认**：`data/upload`）

按用户分组
: 配置槽目录是否应该在用户目录中分组。**（默认：假）**

2.4 逻辑

逻辑负责 URI 的生成和应用限制。它将与插槽分配等相关的所有配置设置分组。

可用属性：

local-only
: 仅允许在本地 XMPP 服务器上拥有帐户的用户使用此组件进行插槽分配。**（默认：真）**

max-file-size
: 以字节为单位设置单个分配槽的最大容量（最大文件容量）。**（默认值：5000）**

port

指定在生成上传和下载 URI 时应该使用的端口。如果未设置，则将使用安全 (HTTPS) 服务器端口（如果可用），在其他情况下使用纯 HTTP。**(默认：未设置)**

protocol

应该使用的协议。**这仅与** `port` **一起使用**。可能的值为：

- http
- https

serverName

在生成的 URI 中用作域部分的服务器名称。**(默认：服务器主机名)**

upload-uri-format

用于生成文件上传 URI 的模板。 **(默认：** `{proto}://{serverName}:{port}/upload/{userJid}/{slotId}/{filename}` **)**

download-uri-format

用于生成文件下载 URI 的模板。**(默认：** `{proto}://{serverName}:{port}/upload/{slotId}/{filename}` **)**

2.4.1 URI 模板格式

{ and } 之间的模板中的每个块都是一个命名部分，在生成插槽的 URI 期间将被属性值替换。

可以使用的块：

proto

协议名称。

serverName

服务器的域名。

port

HTTPS（或 HTTP）服务器正在侦听的端口。

userJid

请求插槽分配的用户的 JID。

domain

请求时隙分配的用户域。

slotId

生成的插槽 ID

filename

要上传的文件名。

备注： `slotId` 和 `filename` 必须是每个 URI 模板的一部分。

警告： 如果配置了多个元数据存储库，则包含 `userJid` 或 `domain` 将加快上传和下载操作期间对插槽 id 的查找。但是，如果将带有包含此部分的 URI 的消息发送给不知道发件人 JID 的收件人（即匿名 MUC 房间的情况），这可能会导致用户 JID 或用户域的泄漏。

2.5 文件上传过期

有时需要删除过期文件以便为新的上传腾出位置。这是由 `expiration` 任务完成的。

可用属性：

expiration-time
: 服务器会将上传的文件保留多长时间。Java Period 格式 的值 **(default: P30D - 30 days)**

period
: 服务器应多久查找一次要删除的过期文件。Java Period 格式 的值 **(default: P1D - 1 天)**

delay
: 自服务器启动到服务器应查找要删除的过期文件之前的时间。Java Period 格式 的值 **（默认值：0)**

limit
: 在 `expiration` 的单次执行期间要删除的最大文件数。**（默认值：10000）**

2.6 例子

2.6.1 复杂配置示例

使用指向 `file_upload` 数据源的 `example.com` 元数据的单独存储库进行配置，自定义上传和下载 URI，最大文件容量设置为 10MB，每 6 小时到期一次，并按用户 jid 对插槽文件夹进行分组。

复杂的配置示例。

```
upload() {
    logic {
        local-only = false
        max-file-size = 10485760
        upload-uri-format = '{proto}://{serverName}:{port}/upload/{userJid}/{slotId}/
↪{filename}'
        download-uri-format = '{proto}://{serverName}:{port}/upload/{domain}/{slotId}/
```

(续下页)

(接上页)

```
↪{filename}'
    }

    expiration {
        period = P6H
    }

    repositoryPool {
        'example.com' () {
            data-source = "file_upload"
        }
    }

    store {
        group-by-user = true
    }
}
```

2.6.2 使用 HA 进行集群的示例配置

集群中的高可用性配置，在 /mnt/shared 有公共存储并且两个服务器都可以使用 upload.example.com

使用 HA 的示例配置。

```
upload() {
    logic {
        upload-uri-format = '{proto}://upload.example.com:{port}/upload/{userJid}/
↪{slotId}/{filename}'
        download-uri-format = '{proto}://upload.example.com:{port}/upload/{domain}/
↪{slotId}/{filename}'
    }

    store {
        path = '/mnt/shared/upload'
    }
}
```

2.7 S3 支持 HTTP 文件上传

默认情况下，Tigase XMPP 服务器附带的 HTTP 文件上传组件将上传的文件本地存储在目录结构中。如果您使用 AWS，最好使用 S3 等更适合此任务且更具弹性的外部服务来存储数据。

备注： 要使此功能正常工作，请确保您正在使用所有必需的依赖项（通过使用 `-dist-max` 包或从 tigase-extras aws 模块 获得）

2.7.1 在 S3 中启用存储

要在 S3 中启用存储，您需要在配置文件中添加以下行：

```
upload () {
    store (class: tigase.extras.http.upload.S3Store, active: true, exportable: true) {
        bucket = 'bucket-name'
    }
}
```

这将启用 HTTP 文件上传组件，并将其配置为与名为 `bucket-name` 的 S3 存储桶一起使用。此存储桶与运行 Tigase XMPP 服务器的 EC2 实例位于同一区域。

> **警告：** 您需要手动创建此 S3 存储桶并允许您的 EC2 实例访问它（读取和写入）。或者，您可以在 `store` 块中添加 `autocreateBucket = true`，这将使 Tigase XMPP 服务器能够在本地 AWS 区域中创建此 S3 存储桶。

如果您希望使用来自另一个 AWS 区域的 S3 存储桶，您可以通过将 `store` 块中的设置 `region` 属性添加到 AWS 区域的 id 来实现，即设置为 `us-west-2` 以使用 `US West (Oregon)` 区域：

```
upload () {
    store (class: tigase.extras.http.upload.S3Store, active: true, exportable: true) {
        bucket = 'bucket-name'
        region = 'us-west-2'
    }
}
```

如果您希望在 Tigase XMPP 服务器的不同安装之间共享同一个 S3 存储桶，您应该为每个安装配置 `store` 的 `bucketKeyPrefix` 属性并使用不同的标识符。这将允许您轻松过滤为每次安装上传的数据，并允许 Tigase XMPP 服务器为您提供每次安装的正确存储使用情况。

```
upload () {
    store (class: tigase.extras.http.upload.S3Store, active: true, exportable: true) {
        bucket = 'bucket-name'
        bucketKeyPrefix = '45252AF'
    }
}
```

S3Store 需要适当的 IAM 策略：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:ListStorageLensConfigurations",
                "s3:GetAccessPoint",
                "s3:PutAccountPublicAccessBlock",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAllMyBuckets",
                "s3:ListAccessPoints",
                "s3:ListJobs",
                "s3:PutStorageLensConfiguration",
                "s3:CreateBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*",
                "arn:aws:s3:*:441807014745:accesspoint/*",
                "arn:aws:s3:*:441807014745:storage-lens/*",
                "arn:aws:s3:*:441807014745:job/*"
            ]
        }
    ]
```

(续下页)

(接上页)

```
}
```

CHAPTER 3

HTTP 服务器

默认情况下，HTTP 服务器实例以 `httpServer` 的形式提供。仅当启用 HTTP API 组件或 HTTP 文件上传组件时，服务器才会处于活动状态并启用。该项目使用嵌入在 Java JDK 中的 HttpServer 提供的 http 服务器的默认实现。

备注： 如果 **不要求**高性能 HTTP 服务器，此实现仅适用于小型安装。如果这不符合您的要求，建议使用 Jetty 作为嵌入式 HTTP 服务器，使用于*Tigase HTTP API - Jetty HTTP Server* 项目。

3.1 依赖项

默认的 HTTP 服务器实现几乎不需要依赖项，因为大多数调用已经嵌入在 JDK 8 中。但是，由于需要一个通用的 API 来处理 HTTP 请求，就像 JDK 和 Jetty 的 HTTP 服务器一样，我们决定使用 HTTP Servlet 3.1 版中的 API。

所需文件可以从 Tigase HTTP API 项目 部分或使用以下链接 servlet-api-3.1.jar 下载

请注意，此文件包含在 Tigase XMPP 服务器的 dist-max、exe 和 jar 安装程序分发中。

3.2 配置属性

可以使用以下所有属性中的任何一个来配置 HTTP 服务器。请注意，这些设置仅适用于 Tigase HTTP API 提供的默认实现。

端口

此属性用于配置 HTTP 服务器上的哪些端口应侦听传入连接。如果未设置，则将使用默认端口 `8080`

连接

它用于对传递给端口的配置进行分组

{port}

对于每个 `{port}`，您可以传递单独的配置。为此，您需要将 `{port}` 替换为端口号，即 `8080`。对于每个端口，您可以传递以下属性：

套接字

设置用于处理传入连接的套接字类型。可接受的值为：

- `plain` - 端口将在普通 HTTP 模式下工作 **（默认）**
- `ssl` - 端口将在 HTTPS 模式下工作

domain

此属性用于配置运行在此端口上的 HTTP 服务器应该使用的 SSL 证书的域名（如果 `socket` 设置为 `ssl`）。如果未设置（或将被省略），则 Tigase XMPP 服务器将尝试对客户端尝试连接的主机使用 SSL 证书。如果该域名没有 SSL 证书，则将使用 Tigase XMPP 服务器的默认 SSL 证书。

3.2.1 嵌入式 HTTP 服务器的附加属性

使用嵌入式 HTTP 服务器，您可以在 `executor` 部分中添加一些附加属性，您可以传递这些属性来调整此 HTTP 服务器。

executor

小节的名称

线程

该属性用于配置用于处理 HTTP 请求的线程数，即 `10`

请求超时

用于设置处理单个 HTTP 请求的超时时间（以毫秒为单位）的属性，即 `30000`

接受超时

用于设置读取 HTTP 请求标头的超时时间（以毫秒为单位）的属性，即 `2000`

3.3 例子

下面是一些用于基于 DSL 的配置格式和基于旧属性的格式的示例。

3.3.1 带有 SSL 证书的端口 8443 上的 HTTPS，例如 example.com

在配置文件 `httpServer` 中相关的配置应该是这样的：

```
httpServer {
    connections {
        8443 () {
            socket = ssl
            domain = 'example.com'
        }
    }
}
```

3.3.2 将端口从 8080 更改为 8081

```
httpServer {
    connections {
        8080 (active: false) {}
        8081 () {}
    }
}
```

从 HTTP 到 HTTPS 的重定向

尽可能多的使用 HTTPS 是有益的，但通常需要添加从 `http` 到 `https` 的重定向。虽然可以使用外部解决方案（额外的 http 服务器，如 nginx 或 apache 或某种具有此类功能的负载均衡器）来完成它，但内置它更方便。

Tigase XMPP 服务器中实现的功能允许指定由目标主机名，可选端口和路径组成的 `redirectUri`。不支持指定任何查询参数。`redirectUri` 支持 `{host}` 变量，该变量可用于在来自原始请求的重定向中保留原始服务器名称，即 `redirectUri = 'https://{host}:8089'` 将请求重定向到同一服务器，但是在端口 8089 上（原始路径 URI 和查询字符串将自动附加到重定向 URL）。

也有以下可能，Tigase XMPP 服务器在其普通套接字端口上处理常规 `http` 请求以及由负载均衡器/代理处理的 `https`，后者终止 HTTPS 流量并使用 `http` 协议转发请求。在这种情况下，无条件请求将导致无限重定向。幸运的是，可以使用 `redirectCondition` 选项指定重定向发生的条件。它必须设置为重定向工作。目前支持以下值（它们应该是不言自明的）：

- `never`,

- http,
- https,
- always

```
httpServer {
    connections {
        8080 () {
            redirectCondition = 'http'
            redirectUri = 'https://{host}:443'
        }
    }
}
```

3.3.3 使用 Jetty HTTP 服务器作为 HTTP 服务器

如前所述，可以使用 Jetty 作为 HTTP 服务器来提高性能。Jetty API 可以以两种形式之一使用：Standalone 和 OSGi。

独立

在这种情况下，Jetty 实例由 Tigase HTTP API 在内部创建和配置。这允许使用与默认 HTTP 服务器配置相同的配置属性。

使用独立的 Jetty HTTP 服务器进行配置。

```
httpServer (class: tigase.http.jetty.JettyStandaloneHttpServer) {
    ...
}
```

HTTP/2 和 Jetty HTTP 服务器

如果在独立模式下使用 Jetty HTTP 服务器，Tigase 使用的 JDK 比 JDK 8 更新，并且 HTTP 服务器被配置为通过加密（ssl 或 tls）连接提供数据，那么 HTTP/2 将是默认启用。

但是，可以通过将加密端口的 use-http2 属性设置为 false 来禁用 HTTP/2，即对于端口 8443：

```
httpServer (class: tigase.http.jetty.JettyStandaloneHttpServer) {
    ...
    '8443' () {
        socket = ssl
        'use-http2' = false
```

(续下页)

(接上页)

```
    }
}
```

OSGi

这只能在 Tigase 在 OSGi 容器内运行时使用。如果这是使用 Tigase HTTP API， 将尝试从 OSGi 容器中检索 Jetty HTTP 服务器并使用它。

备注： Tigase 未配置 Jetty HTTP 服务器实例。我们只会将此实例用于部署。

使用 Jetty HTTP Server 在 OSGi 模式下进行配置。

```
httpServer (class: tigase.http.jetty.JettyOSGiHttpServer) {
    ...
}
```

CHAPTER 4

Rest API

4.1 REST API

Tigase 的 HTTP API 组件使用负责处理传入 HTTP 的 REST 模块和 Groovy 脚本。最终结果是 Tigase 的 REST API。此 API 可能对各种集成场景很有用。

在这些部分中，我们将描述 Tigase HTTP API 提供的基本 REST 端点，并解释创建新自定义端点的基础知识。

具体于特定 Tigase XMPP 服务器模块的其他端点在提供它们的模块的文档中进行了描述。您还可以在 HTTP API 上查看本地 Tigase XMPP 服务器安装上的 `http://localhost:8080/rest/`，它将为您提供安装时可用的 REST 端点的基本使用示例。

有关 REST 模块配置的更多信息，请参阅*REST module* 部分

4.1.1 脚本介绍

HTTP API 组件中的脚本用于处理所有请求。

要向 HTTP API 组件添加新操作，您需要创建一个用 Groovy 编写的脚本，其中将实现扩展 `tigase.http.rest.Handler` 类的类。脚本的 URI 将从脚本文件夹中的文件位置创建。例如，如果带有正则表达式的脚本 `TestHandler` 将被设置为 `/test` 并将被放置在 `scripts/rest/tested` 中，则将使用以下 URI 调用处理程序：`/rest/tested/test`。

属性

如果要扩展类，则需要设置以下属性：

- **regex** - 用于匹配请求 URI 并解析 URI 中嵌入的参数的正则表达式。例如：`/\/([@\/]+)@([@\/]+)/`
- **requiredRole** - 用户的必需角色才能访问此 URI。可用值为：null、"user"和"admin"。如果 `requiredRole` 不为空，则需要身份验证。
- **isAsync** - 如果设置为 true，则可以等待结果，例如等待响应 IQ 节。
- **decodeContent** - 如果设置为 false，则不会解析请求的内容，并且您的脚本将接收 `HttpServletRequest` 的实例来处理传入的内容。

包含闭包的属性

扩展类还应该为以下一个或多个属性设置闭包：execGet、execPut、execPost 和 execDelete，这具体取决于您需要为 URI 支持的 HTTP 操作或操作。**每个闭包都有一个动态参数列表**。下面是传递给闭包的参数列表，它描述了参数列表如何以及何时更改：

1. **service** - 服务接口的实现。这用于访问服务器数据库或发送/接收 XMPP 节。
2. **callback** - 需要调用 `callback` 闭包来返回数据。`callback` 只接受一个 String,byte[],Map 类型的参数。如果数据是 Map 类型，它将根据'Content-Type'标头编码为 JSON 或 XML。
3. **user** - 仅当 `requiredRole` 不为空时才会传递。**在所有其他情况下，此参数将不在参数列表中！**
4. **request** - 只有声明为 `HttpServletRequest` 的实例时才会被传递，并且它将是当前 HTTP 请求的 `HttpServletRequest` 的实例。
5. **content** - 解析的请求内容。如果请求的 Content-Length 为空，此闭包将不在参数列表中。如果 Content-Type 是 XML 或 JSON 作为 Map 返回，否则（或如果 `decodeContent` 设置为 `false`）它将是 `HttpServletRequest` 的实例。
6. **x** - 传递给回调的附加参数是来自与 URI 匹配的正则表达式的组。**组不作为列表传递，而是作为下一个参数添加到参数列表中**。

如果未设置相应 HTTP 操作的属性，则组件将返回 404 HTTP 错误。

访问 beans

可以从实现 REST 处理程序的 groovy 脚本中访问由 Tigase XMPP 服务器管理的 bean。要在 groovy 脚本中实现处理程序类的实现，需要使用 `@Bean` 注释进行注释。在此注解中，您需要传递至少一个参数 `name`，该参数应包含所需的 bean 名称，该处理程序将在 REST 模块内核范围内可用。

有了它，就可以在 `Handler` 实现类的任何字段上使用 `@Inject` 注释来告诉 Tigase 内核注入特定类的实例（或实现特定接口的类的实例）。

有关 Tigase 内核和 bean 的更多详细信息，请查看 Tigase XMPP 服务器开发指南的 `Tigase Kernel` 部分。

实例。

```
@Bean(name = "test-bean", active = true)
class TestHandler
        extends tigase.http.rest.Handler {

    @Inject
    private UserRepository userRepo;

    // implementation of the handler...
}
```

4.1.2 使用示例

检索用户头像

如果在用户 vCard 中设置了头像，则对 url /rest/avatar/admin@test-domain.com 使用 GET 方法的请求将返回用户 admin@test-domain.com 的头像图片，否则将返回 http 错误 404。用户 admin@domain.com 头像的完整 url 示例

```
http://localhost:8080/rest/avatar/admin@domain.com
```

输入这个 url 将执行 GET 请求。您可以在浏览器中使用该网址。

检索可用的临时命令列表

使用 XML 格式

要检索可用的临时命令列表，请使用 GET 方法对 `/rest/adhoc/sess-man@domain.com` 发出请求，其中 `sess-man@domain.com` 您希望查看命令的组件的 jid。例如，在浏览器中输入以下 url：http://localhost:8080/rest/adhoc/sess-man@domain.com 将检索 `sess-man@domain.com` 中可用的所有临时命令的列表。此操作受使用 `HTTP Basic Authentication` 完成的身份验证保护。有效凭据将是此 Tigase XMPP 服务器安装的用户数据库中可用的用户（以 barejid 形式的用户名）。

以下是该请求的示例结果：

```
<items>
  <item>
    <jid>sess-man@domain.com</jid>
    <node>http://jabber.org/protocol/admin#get-active-users</node>
    <name>Get list of active users</name>
  </item>
  <item>
```

(续下页)

(接上页)

```
    <jid>sess-man@domain.com</jid>
    <node>del-script</node>
    <name>Remove command script</name>
  </item>
  <item>
    <jid>sess-man@domain.com</jid>
    <node>add-script</node>
    <name>New command script</name>
  </item>
</items>
```

使用 JSON 格式

要检索 JSON 中可用的即席命令列表，我们需要将 `Content-Type: application/json` 传递给请求的 HTTP 标头或将 `type` 参数集添加到 `application/json`。示例结果如下所示：

```
{
    "items": [
        {
            "jid": "sess-man@domain.com",
            "node": "http://jabber.org/protocol/admin#get-active-users",
            "name": "Get list of active users"
        },
        {
            "jid": "sess-man@domain.com",
            "node": "del-script",
            "name": "Remove command script"
        },
        {
            "jid": "sess-man@domain.com",
            "node": "add-script",
            "name": "New command script"
        }
    ]
}
```

检索命令表单

为了检索具有特定命令的必填字段的表单，您必须从所有可用命令的列表中发送仅带有 `jid` 和 `name` 的 `POST` 请求（使用上述命令返回）

使用 XML

例如，要获取添加 VHost 项目的表单，请使用 `/rest/adhoc/vhost-man@domain.com` 的 `POST` 方法发送以下内容（请求需要使用基本 HTTP 身份验证进行身份验证）：

```
<command>
    <node>comp-repo-item-add</node>
</command>
```

以下是上述请求的示例结果：

```
<command>
    <jid>vhost-man@domain.com</jid>
    <node>comp-repo-item-add</node>
    <fields>
        <item>
            <var>Domain name</var>
            <value/>
        </item>
        <item>
            <var>Enabled</var>
            <type>boolean</type>
            <value>true</value>
        </item>
        <item>
            <var>Anonymous enabled</var>
            <type>boolean</type>
            <value>true</value>
        </item>
        <item>
            <var>In-band registration</var>
            <type>boolean</type>
            <value>true</value>
        </item>
        <item>
            <var>TLS</var>
            <type>fixed</type>
            <value>This installation forces VHost to require TLS. If you need to use␣
↪unencrypted connections set &amp;amp;apos;vhost-tls-required&amp;amp;apos;
```

(续下页)

(接上页)

```
                property to &amp;amp;apos;false&amp;amp;apos; in the installation␣
↪configuration file
            </value>
        </item>
        <item>
            <var>Max users</var>
            <value>0</value>
        </item>
        ...
    </fields>
    <instructions>â??NOTE: Options without value set will use configuration defined␣
↪in 'DEFAULT' VHostâ??</instructions>
</command>
```

使用 JSON

例如，要获取添加 VHost 项目的表单，请使用 `/rest/adhoc/vhost-man@domain.com` 的 `POST` 方法使用 `Content-Type: application/json` 和发送以下内容（请求需要使用基本 HTTP 身份验证进行身份验证）：

```
{
  "command": {
    "node" : "comp-repo-item-add"
  }
}
```

以下是上述请求的示例结果：

```
{
  "command": {
    "jid": "vhost-man@domain.com",
    "node": "comp-repo-item-add",
    "fields": [
      {
        "var": "Domain name",
        "value": null
      },
      {
        "var": "Enabled",
        "type": "boolean",
        "value": "true"
      },
```

(续下页)

(接上页)

```
      {
        "var": "Anonymous enabled",
        "type": "boolean",
        "value": "true"
      },
      {
        "var": "In-band registration",
        "type": "boolean",
        "value": "true"
      },
      {
        "var": "TLS",
        "type": "fixed",
        "value": "This installation forces VHost to require TLS. If you need to use␣
↪unencrypted connections set &apos;vhost-tls-required&apos; property to &apos;false&
↪apos; in the installation configuration file"
      },
      {
        "var": "Max users",
        "value": "0"
      }
      ...
    ],
    "instructions": "NOTE: Options without value set will use configuration defined␣
↪in 'DEFAULT' VHost"
  }
}
```

执行示例临时命令

检索活跃用户列表

使用 XML

要执行命令以获取活动用户列表，请使用 POST 方法对 `/rest/adhoc/sess-man@domain.com` 发出请求，发送以下内容（请求需要使用基本 HTTP 身份验证进行身份验证）：

```
<command>
  <node>http://jabber.org/protocol/admin#get-active-users</node>
  <fields>
    <item>
      <var>domainjid</var>
```

(续下页)

(接上页)

```
      <value>domain.com</value>
    </item>
    <item>
      <var>max_items</var>
      <value>25</value>
    </item>
  </fields>
</command>
```

在这个请求中，我们传递了执行 adhoc 命令所需的所有参数。我们传递了 adhoc 命令的节点和该命令所需字段的值。我们为 “domainjid” 字段传递了” domain.com” 的值，为” max_items” 字段传递了’’ 25’’ 的值。我们还需要将 `Content-Type: text/xml` 传递给请求的 HTTP 标头，或者将 `type` 参数集添加到 `text/xml`。

备注： 如果是多值字段，请使用以下格式：

```
<value>
    <item>first-value</item>
    <item>second-value</item>
</value>
```

以下是上述请求的示例结果：

```
<command>
  <jid>sess-man@domain.com</jid>
  <node>http://jabber.org/protocol/admin#get-active-users</node>
  <fields>
    <item>
      <var>Users: 2</var>
      <label>text-multi</label>
      <value>admin@domain.com</value>
      <value>user1@domain.com</value>
    </item>
  </fields>
</command>
```

使用 JSON

要执行命令以获取 JSON 格式的活动用户，请使用 POST 方法向 /rest/adhoc/sess-man@domain.com 发出请求，发送以下内容（此请求还需要使用基本 HTTP 身份验证进行身份验证）：

```
{
  "command" : {
    "node" : "http://jabber.org/protocol/admin#get-active-users",
    "fields" : [
      {
        "var" : "domainjid",
        "value" : "domain.com"
      },
      {
        "var" : "max_items",
        "value" : "25"
      }
    ]
  }
}
```

在这个请求中，我们传递了执行 adhoc 命令所需的所有参数。我们传递了 adhoc 命令的节点和 adhoc 命令所需字段的值。在这种情况下，我们为’’ domainjid’’ 字段传递了’’ domain.com’’ 的值，为’’ max_items’’ 字段传递了’’ 25’’。

以下是上述请求的示例结果：

```
{
    "command": {
        "jid": "sess-man@domain.com",
        "node": "http://jabber.org/protocol/admin#get-active-users",
        "fields": [
            {
                "var": "Users: 1",
                "label": "text-multi",
                "value": [
                  "admin@domain.com",
                  "user1@domain.com"
                ]
            }
        ]
    }
}
```

结束用户会话

要执行最终用户会话命令，请使用 POST 方法对 `/rest/adhoc/sess-man@domain.com` 发出请求。发送内容的上下文可能因情况而异。例如，它可能需要使用带有管理员凭据的 *Basic HTTP 身份验证*进行身份验证。URL 中的 *sess-man@domain.com* 是会话管理器组件的 JID，通常采用 *sess-man@domain* 的形式，其中 `domain` 是托管域名。

使用 XML

要使用 XML 内容执行命令，您需要将 HTTP 标头 `Content-Type` 设置为 `application/xml`

```
<command>
  <node>http://jabber.org/protocol/admin#end-user-session</node>
  <fields>
    <item>
      <var>accountjids</var>
      <value>
        <item>test@domain.com</item>
      </value>
    </item>
  </fields>
</command>
```

其中 `test@domain.com` 是应该断开的用户的 JID。

结果服务器将返回以下 XML：

```
<command>
  <jid>sess-man@domain.com</jid>
  <node>http://jabber.org/protocol/admin#end-user-session</node>
  <fields>
    <item>
      <var>Notes</var>
      <type>text-multi</type>
      <value>Operation successful for user test@domain.com/resource</value>
     </item>
  </fields>
</command>
```

这将确认具有资源 `resource` 的用户 `test@domain.com` 已连接并已断开连接。

如果用户未连接服务器将返回以下响应：

```
<command>
  <jid>sess-man@domain.com</jid>
```

(续下页)

(接上页)

```
  <node>http://jabber.org/protocol/admin#end-user-session</node>
  <fields />
</command>
```

使用 JSON

要使用 JSON 执行命令，您需要将 HTTP 标头 `Content-Type` 设置为 `application/json`

```
{
  "command" : {
    "node": "http://jabber.org/protocol/admin#end-user-session",
    "fields": [
        {
            "var" : "accountjids",
            "value" : [
                "test@domain.com"
            ]
        }
    ]
  }
}
```

其中 `test@domain.com` 是要断开连接的用户的 JID

结果，服务器将返回以下 JSON：

```
{
  "command" : {
    "jid" : "sess-man@domain.com",
    "node" : "http://jabber.org/protocol/admin#end-user-session",
    "fields" : [
      {
        "var" : "Notes",
        "type" : "text-multi",
        "value" : [
          "Operation successful for user test@domain.com/resource"
        ]
      }
   ]
  }
}
```

这将确认具有资源 `resource` 的用户 `test@domain.com` 已连接并已断开连接。

如果用户未连接服务器将返回以下响应：

```
{
  "command" : {
    "jid" : "sess-man@domain.com",
    "node" : "http://jabber.org/protocol/admin#end-user-session",
    "fields" : []
  }
}
```

虚拟主机/域上的操作

在决定使用 XML 或 JSON 设置相关的 `Content-Type` 标头时，VHosts 上的所有操作都是通过向 `/rest/adhoc/vhost-man@domain.com` 发出 `POST` 请求来完成的（它可能需要使用带有管理员凭据的 *Basic HTTP 身份验证*进行身份验证）。

添加 VHost

添加域是使用带有所有必需和所需字段的 `comp-repo-item-add` 命令完成的（如果缺少某些内容，则将返回要填写的表单）。有关如何检索表单/可用字段的说明，请参阅 检索命令表单。

使用 XML

要使用 XML 内容执行命令，您需要将 HTTP 标头 `Content-Type` 设置为 `application/xml` 和填写的表单（下面是修剪的示例，请参阅 Retrieving command form 了解如何获取完整表单的详细信息）：

备注： 在请求中包含 `command-marker` 是很重要的，否则将返回表单而不添加 VHost。

```
<command>
    <jid>vhost-man@domain.com</jid>
    <node>comp-repo-item-add</node>
    <fields>
        <item>
            <var>Domain name</var>
            <value>my-new-domain.com</value>
        </item>
        <item>
            <var>Enabled</var>
            <value>true</value>
        </item>
```

(续下页)

(接上页)

```
        <item>
            <var>command-marker</var>
            <value>command-marker</value>
        </item>
        ...
    </fields>
</command>
```

如果域添加正确，您将收到 `Operation successful.` 的响应。备注栏：

```
<command>
    <jid>vhost-man@domain.com</jid>
    <node>comp-repo-item-add</node>
    <fields>
        <item>
            <var>Note</var>
            <type>fixed</type>
            <value>Operation successful.</value>
        </item>
    </fields>
</command>
```

使用 JSON

要使用 XML 内容执行命令，您需要将 HTTP 标头 `Content-Type` 设置为 `application/xml` 和填写的表单（下面是修剪的示例，请参阅 Retrieving command form 了解如何获取完整表单的详细信息）：

注意

在请求中包含 `command-marker` 是很重要的，否则将返回表单而不添加 VHost。

```
{
  "command": {
    "jid": "vhost-man@domain.com",
    "node": "comp-repo-item-add",
    "fields": [
      {
        "var": "Domain name",
        "value": "my-new-awesome-domain.com"
      },
      {
        "var": "Enabled",
        "value": "true"
```

(续下页)

(接上页)

```
      },
      {
        "var": "command-marker",
        "value": "command-marker"
      }
      ...
    ]
  }
}
```

如果域添加正确，您将收到 `Operation successful.` 的响应。备注栏：

```
{
  "command": {
    "jid": "vhost-man@domain.com",
    "node": "comp-repo-item-add",
    "fields": [
      {
        "var": "Note",
        "type": "fixed",
        "value": "Operation successful."
      }
    ]
  }
}
```

配置 VHost

使用带有所有必需和所需字段的 `comp-repo-item-update` 命令修改域配置（如果缺少某些内容，将返回表单填写）。有关如何检索表单/可用字段的说明，请参阅 检索命令表单。

使用 XML

要使用 XML 内容执行命令，您需要将 HTTP 标头 `Content-Type` 设置为 `application/xml` 和填写的表单（下面是修剪的示例，请参阅 Retrieving command form 了解如何获取完整表单的详细信息）：

备注： 在请求中包含 `command-marker` 是很重要的（否则将返回表单而不添加 VHost）和 `item-list`，其值设置为正在配置的 VHost 的名称。

```
<command>
    <jid>vhost-man@domain.com</jid>
    <node>comp-repo-item-update</node>
    <fields>
        <item>
            <var>Domain name</var>
            <value>my-vhost.com</value>
        </item>
        <item>
            <var>Enabled</var>
            <value>true</value>
        </item>
        ...
        <item>
            <var>command-marker</var>
            <value>command-marker</value>
        </item>
        <item>
            <var>item-list</var>
            <value>my-vhost.com</value>
        </item>
    </fields>
</command>
```

如果域添加正确，您将收到 `Operation successful.` 的响应。备注栏：

```
<command>
    <jid>vhost-man@domain.com</jid>
    <node>comp-repo-item-update</node>
    <fields>
        <item>
            <var>Note</var>
            <type>fixed</type>
            <value>Operation successful.</value>
        </item>
    </fields>
</command>
```

使用 JSON

要使用 XML 内容执行命令，您需要将 HTTP 标头 `Content-Type` 设置为 `application/xml` 和填写的表单（下面是修剪的示例，请参阅 Retrieving command form 了解如何获取完整表单的详细信息）：

> **注意**
>
> 在请求中包含 `command-marker` 是很重要的（否则将返回表单而不添加 VHost）和 `item-list`，其值设置为正在配置的 VHost 的名称。

```
{
  "command": {
    "jid": "vhost-man@domain.com",
    "node": "comp-repo-item-update",
    "fields": [
      {
        "var": "Domain name",
        "value": "my-domain.com"
      },
      {
        "var": "Enabled",
        "value": "true"
      },
      ...
      {
        "var": "command-marker",
        "value": "command-marker"
      },
      {
        "var": "item-list",
        "value": "my-domain.com"
      }
    ]
  }
}
```

如果域添加正确，您将收到 `Operation successful.` 的响应。备注栏：

```
{
  "command": {
    "jid": "vhost-man@domain.com",
    "node": "comp-repo-item-update",
    "fields": [
      {
        "var": "Note",
```

(续下页)

(接上页)

```
        "type": "fixed",
        "value": "Operation successful."
      }
    ]
  }
}
```

这将确认具有资源 resource 的用户 test@domain.com 已连接并已断开连接。

如果用户未连接服务器将返回以下响应：

```
{
  "command" : {
    "jid" : "sess-man@domain.com",
    "node" : "http://jabber.org/protocol/admin#end-user-session",
    "fields" : []
  }
}
```

发送任何 XMPP 节

XMPP 消息或任何其他 XMPP 节可以使用此 API 通过发送 HTTP POST 请求到（默认情况下）带有序列化 XMPP 节的 http://localhost:8080/rest/stream/?api-key=API_KEY 来发送作为内容，其中 API_KEY 是 HTTP API 的 API 密钥。该键在 etc/config.tdsl 中设置。此外，每个请求都需要通过发送一个有效的管理员 JID 和密码作为 BASIC HTTP 授权方法的用户和密码来进行授权。HTTP 请求的内容应使用 UTF-8 编码，并且 Content-Type 应设置为 application/xml。

处理请求

如果发送的 XMPP 节不包含 from 属性，则 HTTP API 组件将提供它自己的 JID。如果正在发送 iq 节，并且没有设置 from 属性，则接收到的响应将作为 HTTP 响应的内容返回。成功的请求将返回 HTTP 响应代码 200。

例子

发送一条 XMPP 消息，从集合到 HTTP API 组件到完整的 JID。

数据需要作为 HTTP POST 请求内容发送到 HTTP API 组件的 URL /rest/stream/?api-key=API_KEY 以将消息 *Example message 1* 传递到 *test@example.com/resource-1*。

```
<message xmlns="jabber:client" type="chat" to="test@example.com/resource-1">
    <body>Example message 1</body>
</message>
```

用 from 发送一个 XMPP 消息，将 HTTP API 组件设置为裸 JID。

数据需要作为 HTTP POST 请求内容发送到 HTTP API 组件的 /rest/stream/?api-key=API_KEY URL，以将消息 *Example message 2* 传递到 *test@example.com*。

```
<message xmlns="jabber:client" type="chat" to="test@example.com">
    <body>Example message 2</body>
</message>
```

发送 XMPP 消息，其中 from 设置为指定的 JID 和收件人的完整 JID。

数据需要作为 HTTP POST 请求内容发送到 HTTP API 组件的 URL /rest/stream/?api-key=API_KEY 以将消息 *示例消息 3* 传递到 *test@example.com/resource -1* 并将消息的发件人设置为 *sender@example.com*。

```
<message xmlns="jabber:client" type="chat" from="sender@example.com" to="test@example.
↪com/resource-1">
    <body>Example message 1</body>
</message>
```

设置 XMPP 用户状态

默认情况下，当客户端断开连接时，XMPP 用户显示为不可用。然而，在某些情况下，我们可能希望向用户展示一个活动，并设置了一些特定的存在。为了控制不可用 XMPP 用户的存在，我们可以使用此功能。

下面显示的示例内容需要发送到（默认情况下）http://localhost:8080/rest/user/{user-jid}/status?api-key=API_KEY，其中：

- API_KEY 是 HTTP API 的 API 密钥
- {user-jid} 是您要为其设置状态的用户的裸 jid。

小技巧： 您可以在 /status 部分之后添加 /{resource} 到 URL 中，其中 {resource} 是要为其设置存在的资源的名称。

警告

您需要将 'user-status-endpoint@http.{clusterNode}' 添加到受信任的 jid 列表中，以允许 UserStatusEndpoint 模块与 Tigase XMPP 服务器正确集成。

使用 XML

要设置用户状态，您需要将 HTTP 标头 `Content-Type` 设置为 `application/xml`

```
<command>
    <available>true</available>
    <priority>-1</priority>
    <show>xa</show>
    <status>On the phone</status>
</command>
```

其中：

- `available` - 或许：
  - `true` - 用户可用/已连接 **（默认）**
  - `false` - 用户不可用/断开连接
- `priority` - 存在优先级的整数。（*应始终设置为负值以确保不丢弃消息*）**（默认值：-1）**
- `show` - 可能是 `presence/show` 元素值之一 **（可选）**
  - `chat`
  - `away`
  - `xa`
  - `dnd`
- `status` - 应该作为存在状态消息所发送的消息 **（可选）**

结果服务器将返回以下 XML：

```
<status>
  <user>test@domain.com/tigase-external</user>
  <available>true</available>
  <priority>priority</priority>
  <show>xa</show>
  <status>On the phone</status>
  <success>true</success>
</status>
```

这将确认具有资源 `tigase-external` 的用户 `test@domain.com` 已更改其存在（查找 `success` 元素值）。

使用 JSON

要设置用户状态，您需要将 HTTP 标头 `Content-Type` 设置为 `application/json`

```
{
  "available": "true",
  "priority": "-1",
  "show": "xa",
  "status": "On the phone"
}
```

其中：

- `available` - 或许：
  - `true` - 用户可用/已连接 **（默认）**
  - `false` - 用户不可用/断开连接
- `priority` - 存在优先级的整数。（*应始终设置为负值以确保不丢弃消息*）**（默认值：-1）**
- `show` - 可能是 `presence/show` 元素值之一 **（可选）**
  - `chat`
  - `away`
  - `xa`
  - `dnd`
- `status` - 应该作为存在状态消息所发送的消息 **（可选）**

结果，服务器将返回以下 JSON：

```
{
  "status": {
    "user": "test@domain.com/tigase-external",
    "available": "true",
    "priority": "-1",
    "show": "xa",
    "status": "On the phone",
    "success": true
  }
}
```

这将确认具有资源 `tigase-external` 的用户 `test@domain.com` 已更改其存在（查找 `success` 元素值）。

4.1.3 BOSH HTTP 预绑定

Bosh (HTTP) 预绑定

绑定用户会话是通过使用 HTTP POST 方法为 `/rest/adhoc/bosh@domain.com` 发送具有以下内容的请求来完成的：

备注： 请求需要使用基本 HTTP 身份验证进行身份验证

```
<command>
  <node>pre-bind-bosh-session</node>
  <fields>
    <item>
      <var>from</var>
      <value>user_jid@domain/resource</value>
    </item>
    <item>
      <var>hold</var>
      <value>1</value>
    </item>
    <item>
      <var>wait</var>
      <value>60</value>
    </item>
  </fields>
</command>
```

配置

可以调整以下参数：

- **from** 这将是用户的 JID。您可以更改由 `from` 变量标识的项目的 `<value/>` 节点；这可以是 FullJID 或 BareJID。在后一种情况下，将为正在绑定的会话生成一个随机资源。
- **hold** 值。通过更改由 `hold` 变量标识的项目的 `<value/>` 节点的值。该值与 XEP-0124: Session Creation Response 中指定的 `hold` 属性匹配
- **wait** 值。通过更改由 `wait` 变量标识的项目的 `<value/>` 节点的值。该值与 XEP-0124: Session Creation Response 中指定的 `wait` 属性匹配

作为响应，将收到一个 XML，其结果包含额外的可用会话和 RID，其可用于客户端附加到会话，例如：

```
<command>
  <jid>bosh@vhost</jid>
```

(续下页)

(接上页)

```
  <node>pre-bind-bosh-session</node>
  <fields>
    <item>
      <var>from</var>
      <label>jid-single</label>
      <value>user_jid@domain/resource</value>
    </item>
    <item>
      <var>hostname</var>
      <label>jid-single</label>
      <value>node_hostname</value>
    </item>
    <item>
      <var>rid</var>
      <label>text-single</label>
      <value>9929332</value>
    </item>
    <item>
      <var>sid</var>
      <label>text-single</label>
      <value>3f1b6e70-8528-44bb-8f23-77e7c4a8cf1a</value>
    </item>
    <item>
      <var>hold</var>
      <label>text-single</label>
      <value>1</value>
    </item>
    <item>
      <var>wait</var>
      <label>text-single</label>
      <value>60</value>
    </item>
  </fields>
</command>
```

例如，将上述 XML 请求存储在 prebind 文件中，可以使用 $curl 执行请求：

```
>curl -X POST -d @prebind http://admin%40domain:pass@domain:8080/rest/adhoc/
↪bosh@domain --header "Content-Type:text/xml"
```

使用 JSON

要执行命令以 JSON 格式预绑定 BOSH 会话，请使用 POST 方法向 `/rest/adhoc/bosh@domain.com` 发出请求，发送以下内容：

```
{
  "command" : {
    "node" : "pre-bind-bosh-session"",
    "fields" : [
      {
        "var" : "from",
        "value" : "user_jid@domain/resource"
      },
      {
        "var" : "hold",
        "value" : "1"
      },
      {
        "var" : "wait",
        "value" : "60"
      }
    ]
  }
}
```

此示例以 XML 格式复制上述相同的请求。