BigFix
MCP Server Guide

Special notice

Before using this information and the product it supports, read the information in Notices (on page 31).

Edition notice

This edition applies to BigFix version 11 and to all subsequent releases and modifications until otherwise indicated in new editions.

Chapter 1. Introduction

Learn how the BigFix Platform MCP Server extends the capabilities of BigFix by allowing compatible AI clients to use BigFix functionalities through the Model Context Protocol (MCP).

The BigFix MCP Server enables AI agents and clients to interact with BigFix endpoints and resources using natural language commands. This guide explains its features, benefits, and how it allows any AI client to seamlessly connect and interact with a BigFix deployment.

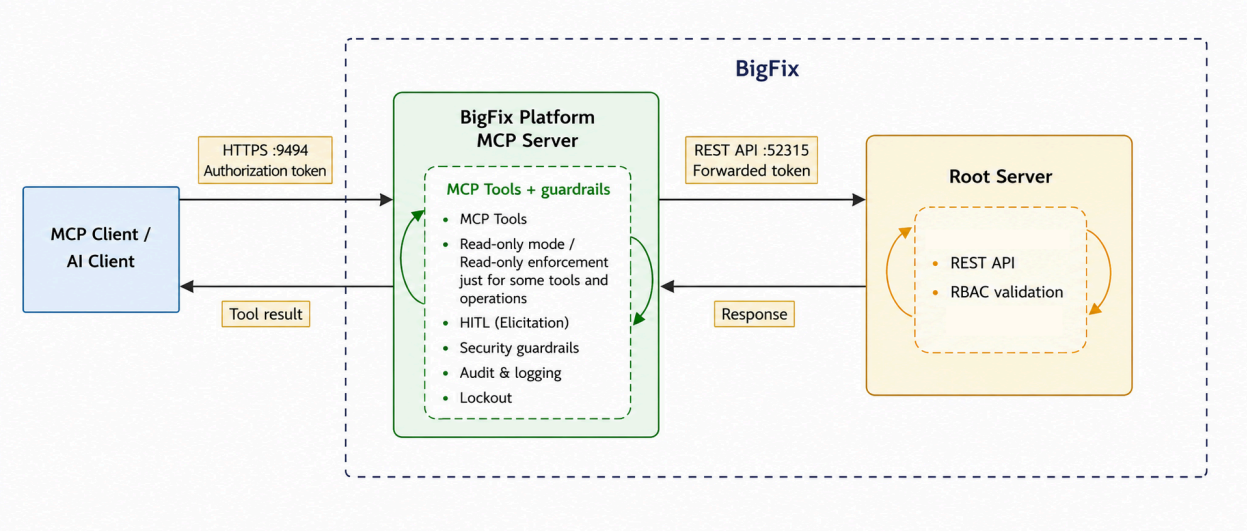
The Model Context Protocol (MCP) functions as an open standard enabling a secure, structured bridge between large language models (LLMs) and real-world tools, APIs, and endpoint repositories. In the context of the BigFix ecosystem, integrating an MCP server allows external agentic AI workflows to natively "understand" and command endpoint data without needing bespoke API connectors for every task.

The BigFix Platform MCP Server is a managed, standalone BigFix integration service. It runs in streamable HTTP mode over HTTPS for network-accessible MCP clients and acts as a token-forwarding gateway. The Authorization header received from the MCP client is forwarded directly to the BigFix Root Server REST API so that each request is evaluated with the BigFix identity represented by the supplied token.

The MCP Server does not create a shared privileged BigFix identity and does not bypass BigFix permissions. The BigFix Root Server validates the token, identifies the operator, applies role-based access control (RBAC), and decides whether the requested operation is permitted.

To safeguard operations further, administrators can enforce granular, policy-based guardrails natively within the MCP Server configuration, such as read-only mode and human in the loop (HITL).

The following diagram displays the high-level BigFix Platform MCP Server architecture and trust boundaries.



Important: The MCP Server uses the client-supplied BigFix REST API token. Human approval and MCP security controls do not grant additional BigFix permissions; Root Server RBAC remains authoritative.

The BigFix Platform MCP Server depends on REST API bearer-token support introduced in BigFix Version 11.0.6. For more details, see Configuring bearer token authentication.

TLS is supported natively. If a listener certificate and matching private key are configured, the MCP Server uses them. Otherwise, the service can generate a self-signed certificate in its workspace.

Chapter 2. Supported scenarios

The current BigFix Platform MCP Server version supports the following scenarios:

- Action lifecycle management, including listing actions, retrieving action details and status, inspecting downloads, retrying downloads, stopping actions, creating actions, and deleting actions.
- Custom action authoring.
- Deployment of existing Fixlets or Tasks through action creation.
- Computer discovery, including listing managed computers and retrieving details for a specific computer.
- Fixlet discovery, including listing Fixlets in a site and retrieving complete Fixlet details by site and ID.
- Session Relevance evaluation through BigFix Web Reports or BigFix Explorer, provided that one of these services is installed and reachable from the Root Server.
- Management operations for the tool families listed in the **Currently implemented tools** section described in Using BigFix Platform MCP Server *(on page 17)*.

Chapter 3. Security Architecture Flow

The MCP Server runs as a standalone service and acts as a controlled bridge between an external MCP client and the BigFix Root Server REST API.

1. An MCP client connects to the MCP Server through the streamable HTTP endpoint, normally over HTTPS on port 9494.
2. The MCP client sends a BigFix REST API token in the HTTP `Authorization` header..
3. The MCP Server extracts the `Authorization` value and stores it only in the request context needed for the current call.
4. The selected MCP tool validates its inputs and builds the corresponding BigFix REST API request.
5. Before write-capable operations are dispatched, the MCP Server applies its effective read-only policy and, when enabled, Human-in-the-Loop approval.
6. The MCP Server forwards the same `Authorization` value to the BigFix Root Server REST API, normally on port 52315.
7. The Root Server validates the token, identifies the BigFix operator, applies RBAC, processes the request, and returns the response.
8. The MCP Server formats the result and returns it to the MCP client.

Security boundary: The MCP client is external. The MCP Server and Root Server are BigFix-managed components, but they remain separate services connected through HTTPS. The MCP Server is not an in-process extension of the Root Server.

Chapter 4. Requirements

Before installing the BigFix Platform MCP Server, verify the following requirements.

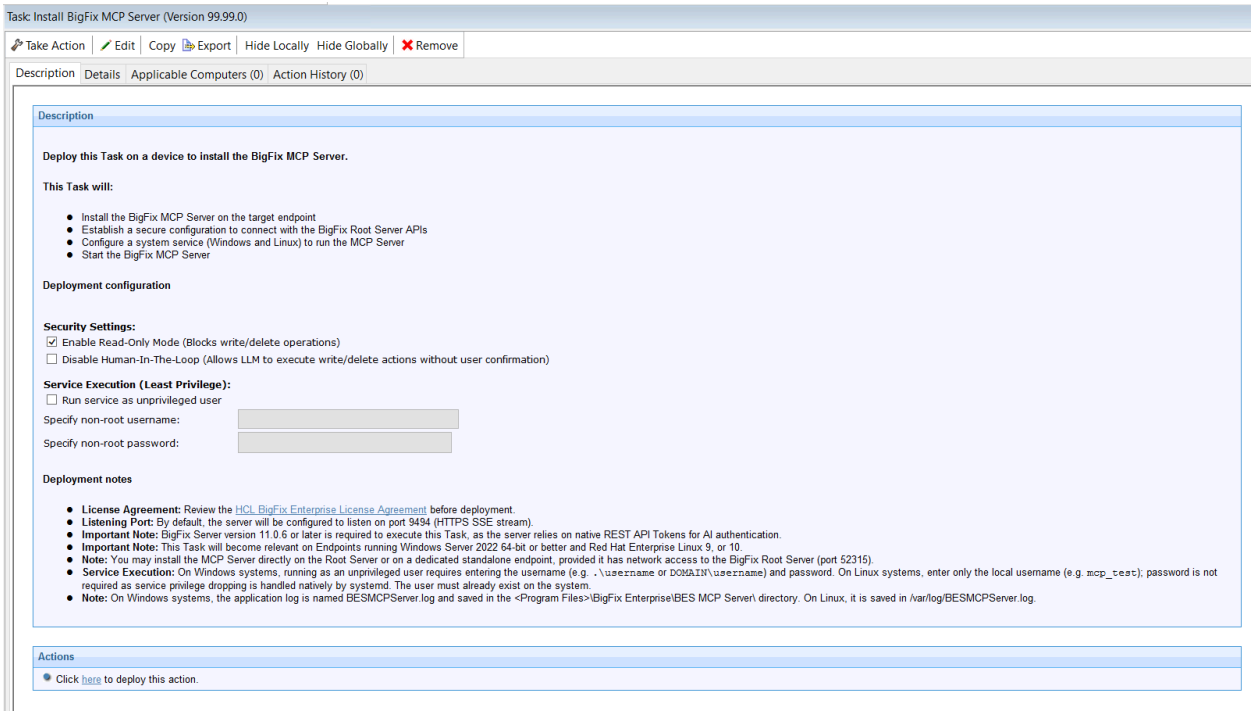
Requirement	Details
BigFix Platform	BigFix Platform Version 11.0.6 or later.
Installation target	The MCP Server can be installed on the BigFix Root Server or on a separate endpoint.
Operating systems	Windows Server 2022 or later, 64-bit; Red Hat Enterprise Linux 9 or 10, 64-bit.
Client-to-MCP connectivity	MCP clients must be able to reach the MCP Server listener, normally TCP port 9494.
MCP-to-Root connectivity	The MCP Server endpoint must be able to reach the Root Server REST API, normally TCP port 52315.
BigFix authentication	A valid BigFix REST API bearer token for each user or client session.
MCP client capability	A compatible client that supports remote streamable HTTP MCP servers and custom request headers. Write operations with server-side HITL also require MCP Form Elicitation support.
TLS trust	The MCP client must trust the MCP Server listener certificate. The MCP Server must trust the certificate chain presented by the Root Server.

Chapter 5. Installing and configuring BigFix Platform MCP Server

The recommended installation method is the BigFix Installation Fixlet. The Fixlet installs the service and creates its initial configuration. After installation, administrators can review or modify the configuration file and restart the service to apply the changes.

Installation

Installation is currently supported as a fresh installation only. The Installation Fixlet, named **Install BigFix MCP Server**, is relevant only when the `bes-mcp-server` service does not already exist. It is not currently an upgrade or repair workflow.



Fixlet option	What it controls
Run service as unprivileged user	Selects the operating-system account used to run the MCP Server service.
Read-only mode	Enables the server-side blocking of write-capable operations.

Fixlet option	What it controls
Human-in-the-Loop / disable HITL	Determines whether the supported write operations require explicit user confirmation.

Installation Fixlet options

Run service as unprivileged user

The Installation Fixlet can run the MCP Server service with a dedicated account that is not root on Linux and is not a local Administrator on Windows. The installation action itself still requires the privileges needed to install a service, create directories, and apply file permissions.

Windows: Select **Run service as unprivileged user** and provide the account in DOMAIN \username or MACHINE\username format. Omitting the domain or machine prefix can cause service installation to fail. The Fixlet adjusts ownership or access control on the MCP Server installation and data directories for the selected account.

Linux: Select **Run service as unprivileged user** and provide the existing account that will run the service. The Fixlet adjusts ownership of the required MCP Server directories.

Minimum runtime access for the service account

The service account does not need BigFix Console privileges merely because it runs the MCP Server. BigFix authorization is performed by the Root Server using the bearer token supplied with each MCP request. At operating-system level, the service account must be able to:

- Start and run the BES MCP Server service.
- Read and execute the MCP Server binary and required runtime files.
- Read and write the configured workspace and data directory.
- Create and append the configured application and audit log files.
- Read the Root Server CA certificate configured by the Installation Fixlet.
- Read the listener certificate and private key when custom MCP listener TLS files are configured.

- Open the configured listener port, normally 9494.
- Establish outbound HTTPS connections to the Root Server REST API, normally on port 52315.

Windows account rights: The account must have the **Log on as a service** permission. Confirm in the delivered Fixlet whether this right is assigned automatically; if it is not, it must be granted before the service starts.

Read-only mode option

When the Fixlet enables read-only mode, it writes the corresponding server configuration so that write capable operations are blocked by the MCP Server before they are sent to the Root Server. This is an execution-time control, not only guidance for the AI model. Choose read-only mode for evaluation, discovery, reporting, or environments where the MCP client must not create, update, retry, stop, or delete BigFix objects. For the complete server/client decision logic, see **Read-only mode** described in Using BigFix Platform MCP Server (on page 17).

Human-in-the-Loop option

The HITL option controls whether supported write-capable operations require an explicit user decision through MCP Form Elicitation. In the configuration, disable_hitl: false means HITL is enabled. disable_hitl: true disables the server-configured confirmation step. Before enabling write operations with HITL, verify that the selected MCP client advertises Form Elicitation support. If the client does not support this capability, the MCP Server blocks the write operation rather than executing it without confirmation.

Trust validation between the MCP Server and the BigFix Root Server

This Fixlet field is used for Root Server trust. It specifies an absolute path on the installation target to a CA certificate or certificate file that allows the MCP Server to validate the TLS certificate presented by the Root Server REST API on port 52315. This field is not the certificate and private key presented by the MCP Server to MCP clients on port 9494.

Configuration

After the installation, a new service is available to manage the BES MCP Server process and starts automatically. The Installation Fixlet creates the initial configuration. Administrators

can then review or customize the configuration by editing the configuration file and restarting the service.

Default configuration paths:

- **Windows**: `C:\Program Files\BigFix Enterprise\BES MCP Server\data\config.yaml`
- **Linux**: `/var/opt/BESMCPServer/data/config.yaml`

Restart required: Any change to config.yaml must be followed by a restart of the `bes-mcp-server` service.

Listener settings

Field	Description	Default / notes
mcp_server.scheme	Protocol used by the MCP Server listener.	Default: https. HTTP should be limited to local development or tightly controlled internal environments.
mcp_server.port	Listening port for MCP clients.	Default: 9494. Clients must be able to reach this port.
mcp_server.tls.cert_path	Optional path to the TLS certificate presented by the MCP Server listener.	When supplied together with key_path, the service uses this certificate instead of generating a self-signed certificate.
mcp_server.tls.key_path	Optional path to the private key matching the listener certificate.	The service account must be able to read this file.

Security runtime settings

Field	Description	Default / notes
mcp_server.read_only	Enables server-side read-only enforcement.	Default: true. Write-capable operations are blocked according to generated tool metadata.
mcp_server.disable_hitl	Controls server-side Human-in the-Loop confirmation.	Default: false. False means HITL is enabled; true means the server-configured HITL step is disabled.

Root Server connection settings

Field	Description	Default / notes
bigfix.url	BigFix Root Server REST API base URL.	Default pattern: `https://<root_server>:52315`
bigfix.ca_cert_path	Certificate or CA material used to validate the TLS certificate presented by the Root Server.	Normally populated by the Installation Fixlet. The service fails closed if the trust chain cannot be validated.

Logging settings

Field	Description	Default / notes
log.file_path	Path of the MCP Server application log.	Platform-specific default configured by the installer.
log.max_size_mb	Maximum size in MB of each log file before rotation.	Default: 100.
log.max_backups	Maximum number of rotated log files to retain.	Default: 3.

Field	Description	Default / notes
log.max_age_days	Maximum age in days of retained log files.	Default: 28.
log.compress	Controls compression of rotated log files.	Default: false.

A representative configuration is:

```
mcp_server:
  scheme: "https"
  port: "9494"
  read_only: true
  read_only_tool: []
  disable_hitl: false

tls:
  cert_path: "/path/to/mcp-listener.crt"
  key_path: "/path/to/mcp-listener.key"

bigfix:
  url: "https://root-server.example.com:52315"
  ca_cert_path: "/path/to/root-server-ca.crt"

log:
  file_path: "/path/to/log/BESMCPServer.log"
  max_size_mb: 100
  max_backups: 3
  max_age_days: 28
  compress: false
```

TLS and Certificates

There are two separate TLS directions. They use different certificates and must not be described as one generic custom-certificate setting.

Connection	Certificate being validated	Configuration
MCP Client -> MCP Server, normally port 9494	Certificate presented by the MCP Server listener.	mcp_server.tls.cert_path and mcp_server.tls.key_-path, or an automatically generated self signed certificate.
MCP Server -> Root Server, normally port 52315	Certificate presented by the Root Server REST API.	bigfix.ca_cert_path is initialized by the Installation Fixlet in config.yaml.

Root Server trust

The MCP Server mandates TLS certificate validation for communication with the BigFix Root Server. By default, the Installation Fixlet locates the Root Server certificate from the BigFix Client installation, copies it into the MCP Server data directory, and configures bigfix.ca_cert_path. The calculated REST API port is 52315 by default.

MCP Server listener certificate

The certificate presented to MCP clients is configured independently. There are two options:

- Provide a certificate and matching private key in mcp_server.tls.cert_path and mcp_server.tls.key_path.
- Leave both fields unset and allow the service to generate a self-signed certificate automatically.

The hostname or IP used in the MCP client URL must be included in the certificate Subject Alternative Name. The certificate or its issuing CA must be trusted in the operating-system environment where the MCP client opens the connection. For VS Code Remote SSH, WSL,

Codespaces, and development containers, trust must also be installed in the remote environment.

Logging

Application logging is platform-specific:

- **Windows:** BESMCPServer.log under the configured BigFix Enterprise installation location.
- **Linux:** /var/log/BESMCPServer.log by default.

The MCP Server also produces a formatted audit log named mcp_audit.log in the MCP Server data directory. The audit log records tool execution metadata, Root Server response sizes and execution times, read-only and HITL security decisions, and lockout-related events.

Sensitive data: Raw authorization tokens are never logged. Sensitive keys are masked and unusually large payloads are truncated.

When the MCP Server is removed with the **Uninstall BigFix MCP Server** Fixlet, the configuration directory is deleted, while the .log files are preserved in the installation folder for troubleshooting purposes, subject to the behavior of the delivered Fixlet.

Chapter 6. Using BigFix Platform MCP Server

Configuring the client

The BigFix Platform MCP Server can be used by MCP-compatible AI clients that support remote streamable HTTP MCP servers and custom request headers. GitHub Copilot in Visual Studio Code is one validated client configuration.

For write operations with HITL enabled, the client must also advertise MCP Form Elicitation support. A client that lacks this capability can still use permitted read operations, but HITL-protected write operations are blocked.

Example of `mcp.json` configuration:

```
{
  "inputs": [
    {
      "type": "promptString",
      "id": "bigfix-token",
      "description": "BigFix REST API token",
      "password": true
    }
  ],
  "servers": {
    "bigfixMcp": {
      "type": "http",
      "url": "https://<mcp-host>:9494",

      "headers": {
        "Authorization": "Bearer ${input:bigfix-token}",
        "X-Bes-Mcp-Read-Only": "true",
        "X-Bes-Mcp-Disable-Hitl": "false"
      }
    }
```

```
  }
}
```

After adding the configuration:

1. Start or reload the MCP server entry from Visual Studio Code.
2. Confirm certificate trust when prompted and ensure the listener certificate is trusted by the environment opening the connection.
3. Use MCP: List Servers to verify that the server is connected.
4. Use Copilot Chat in Agent mode so that MCP tools are available to the model.
5. For Copilot Business or Copilot Enterprise, verify that the organizational policy allows MCP server usage.

Official references:

GitHub Copilot MCP setup

VS Code MCP configuration reference

Currently implemented tools

The following tool families are integrated through the OpenAPI specification and bridge MCP requests to the BigFix Root Server:

- Action Management
- Analysis Management
- Baseline Management
- Computer Management
- Fixlet Management
- Identity and access management, including identity providers, LDAP, Operators, and Roles
- Query and Session operations
- Site Management
- Task Management

The available operations and required parameters are exposed through the MCP tool schemas. Users can ask the connected AI client to describe a tool family, but product documentation should still provide examples for important operational workflows and should not rely exclusively on the model to explain the product.

Note: To be able to call any tool, included the "list tools", you must always include the token of the operator inside the Authorization header. Otherwise, you will be blocked by the MCP server.

Security Guardrails

Read-only mode

Read-only mode is enforced by the MCP Server at execution time. It is not only an instruction in the tool description or a suggestion to the AI model.

The server compares the requested operation with generated tool metadata. Operations classified as create, update, retry, stop, delete, or otherwise write-capable are blocked before any request is dispatched to the Root Server when effective read-only mode is active.

Read-only is evaluated from both the server configuration and the client header:

Server mcp_-server.read_only	Client X-Bes-Mcp-Read-Only	Effective behavior
true	Any value or missing	Read-only. Write-capable operations are blocked.
false	true	Read-only. Write-capable operations are blocked.
false	missing or invalid	Read-only by restrictive default.
false	false	Write-capable operations may proceed to HITL and Root Server RBAC.

The read-only mode can be enforced either for all tools and operations by correctly setting the parameter "read_only" in the config.yaml, or, you can disable the "read_only" in the config.yaml and writing the tool or list of tools you want to disable inside the parameter read_only_tools: [] separated by a comma.

The tools you can disable are the following:

- `manage_bigfix_action`
- `manage_bigfix_analyses`
- `manage_bigfix_baselines`
- `manage_bigfix_computers`
- `manage_bigfix_fixlets`
- `manage_bigfix_idp`
- `manage_bigfix_ldap`
- `manage_bigfix_operators`
- `manage_bigfix_query`
- `manage_bigfix_roles`
- `manage_bigfix_session`
- `manage_bigfix_sites`
- `manage_bigfix_tasks`

Disabling read-only does not bypass Human-in-the-Loop or Root Server RBAC. It only allows a write-capable request to continue to the next security controls.

Human-in-the-Loop mode

The MCP Server implements server-side Human-in-the-Loop enforcement for supported write-capable operations by using MCP Form Elicitation. The server stages the request before execution and asks the user to review and authorize that exact request.

Note: Not all AI clients support MCP Form Elicitation. Therefore, ensure that the AI client that you are using supports this feature.

HITL is an execution-time security control. It does not rely on an AI model remembering to ask for confirmation in the chat, and it does not replace BigFix Root Server authentication or

RBAC. After a valid approval, the original caller token is still forwarded and the Root Server makes the final authorization decision.

The protected write flow is as follows:

1. The MCP Server classifies the requested operation as write-capable.
2. Read-only policy is applied first. If the operation is blocked, no HITL request is created.
3. The server stages an exact copy of the tool input and calculates a SHA-256 binding over the tool name, operation, and exact JSON payload bytes.
4. The approval is bound to the current MCP session and a fingerprint of the caller.
5. The server creates a pending approval with a five-minute lifetime.
6. A Form Elicitation request displays the review data and asks the user to select Allow, Deny, or Cancel.
7. Before execution, the server revalidates the session, caller, payload hash, approval state, expiry, and single-use conditions.
8. Only a valid Allow decision consumes the approval and executes the staged payload once.

The review message contains a JSON object with the following fields:

```
{
  "tool": "manage_bigfix_action",
  "operation": "deleteAction",
  "approval_id": "hitl_<generated-id>",
  "payload_sha256": "<sha-256>",
  "exact_staged_request": {
    "operation": "deleteAction",
    "action_id": 1123
  }
}
```

The SHA-256 value binds the authorization to the exact staged request bytes. Any changed request requires a new approval

Decision handling is as follows:

Decision	Meaning	Server behavior
Allow	Authorize this exact staged request.	Validate and consume the approval, then execute the staged payload once.
Deny	Explicitly reject the request.	Mark the request as denied. Nothing is sent to the Root Server.
Cancel	Close the confirmation without authorization.	Cancel or deny the approval. Nothing is sent to the Root Server.
Timeout / Invalid response	No valid authorization was received.	Expire or reject the approval and fail closed

Configuration and runtime header:

Control	Value	Effect
mcp_server.disable_hitl	false	Server-side HITL is enabled for eligible write operations.
mcp_server.disable_hitl	true	Server configuration disables HITL.
X-Bes-Mcp-Disable-Hitl	false or omitted	The client does not request an HITL bypass.
X-Bes-Mcp-Disable-Hitl	true	The client requests an HITL bypass. Treat this as a trusted client control.

Effective behavior: HITL remains active only when the server setting does not disable it and the client does not send the disable header as true. BigFix RBAC always remains authoritative.

Client rendering and compatibility:

- The MCP Server controls the review message, the form question, the field description, the decision enum, and the default decision.
- The current form uses a string decision with Allow, Deny, and Cancel. Deny is the safe default.
- The MCP client controls native interface labels such as Submit, Respond, Accept, Decline, or Cancel. The server cannot rename client-native buttons.
- The decision explanation should appear only once, in the description beneath the form question. The review message should contain only the operation review and staged request details.
- If the client does not advertise compatible Form Elicitation support, the MCP Server blocks the protected write operation instead of continuing without approval

Audit and troubleshooting:

HITL lifecycle and execution outcomes are written to the MCP audit trail. Relevant events include:

HITL_PENDING	HITL_APPROVED
HITL_DENIED	HITL_EXPIRED
HITL_EXECUTED	HITL_BYPASSED
HITL_BLOCKED	HITL_STATE_ERROR

Automatic IP-based lockout mechanism

If a client exceeds the configured number of consecutive failed requests, which is 5, the source IP address is temporarily locked out for 5 minutes. While the lockout is active, further requests from that IP address are rejected immediately by the MCP Server.

After the lockout period expires, the client can attempt a new request. Lockout and related security events are recorded in the MCP Server logs.

Chapter 7. Troubleshooting BigFix Platform MCP Server

VSCode returns "TypeError: fetch failed" on MCP client startup

A fetch failure when connecting to `https://<mcp-host>:9494` is commonly caused by listener certificate name mismatch or missing certificate trust in the environment where VS Code opens the MCP connection.

A self-signed certificate works only when both conditions are met:

- The MCP URL uses a hostname or IP address present in the certificate Subject Alternative Name.
- The certificate or its issuing CA is trusted by the operating-system environment where the MCP client runs.

When HTTPS is enabled and no custom listener certificate is configured, the MCP Server generates a self signed certificate. The generated certificate includes localhost and 127.0.0.1 and can include the configured mcp_server.server value.

1. Use an MCP URL that matches the certificate, such as localhost, 127.0.0.1, or the configured host/IP included in the SAN.
2. Trust the certificate in the correct operating-system trust store.
3. Reload VS Code or restart the remote extension host after adding trust.
4. If the service regenerated the certificate, trust the new certificate because its fingerprint changed.
5. For Remote SSH, WSL, Codespaces, or development containers, install trust in the remote environment rather than only on the local workstation.

HTTP alternative: HTTP avoids certificate validation but provides no encryption or server identity validation. Use it only for local development or tightly controlled internal environments.

Example 1: HTTPS with a self-signed listener certificate

```
mcp_server:
  scheme: "https"
  server: "10.X.X.X"
  port: "9494"
  read_only: true
  disable_hitl: false

bigfix:
  url: "https://my-bigfix-server:52315"
  ca_cert_path: "C:/path/to/root-server-ca.crt"

log:
  file_path: "C:/Program Files/BigFix Enterprise/BES MCP
 Server/BESMCPServer.log"
{
  "servers": {
    "bigfixMcp": {
      "type": "http",
      "url": "https://10.X.X.X:9494",
      "headers": {
        "Authorization": "Bearer ${input:bigfix-token}",
        "X-Bes-Mcp-Read-Only": "true",
        "X-Bes-Mcp-Disable-Hitl": "false"
      }
    }
  },
  "inputs": [
    {
      "type": "promptString",
      "id": "bigfix-token",
      "description": "BigFix REST API token",
      "password": true
```

```
    }
  ]
}
```

Example 2: HTTPS with a custom listener certificate

```
mcp_server:
  scheme: "https"
  server: "mcp-server.example.com"
  port: "9494"
  read_only: true
  disable_hitl: false
  tls:
    cert_path: "C:/Program Files/BigFix Enterprise/BES MCP
 Server/certs/server.crt"
    key_path: "C:/Program Files/BigFix Enterprise/BES MCP
 Server/certs/server.key"

bigfix:
  url: "https://my-bigfix-server:52315"
  ca_cert_path: "C:/path/to/root-server-ca.crt"

log:
  file_path: "C:/Program Files/BigFix Enterprise/BES MCP
 Server/BESMCPServer.log"
```

Example 3: HTTP without TLS

```
mcp_server:
  scheme: "http"
  server: "10.X.X.X"
  port: "9494"
  read_only: true
  disable_hitl: false
```

```
bigfix:
  url: "https://my-bigfix-server:52315"
  ca_cert_path: "C:/path/to/root-server-ca.crt"

log:
  file_path: "C:/Program Files/BigFix Enterprise/BES MCP
 Server/BESMCPServer.log"
```

No listener TLS: No certificate trust is required for the client-to-MCP connection, but traffic is not encrypted and the client cannot validate the identity of the MCP Server.

Log evidence examples

```
2026-07-31 15:22:55.359 [info] Connection state: Running
2026-07-31 15:22:55.459 [info] Connection state: Error
Error sending message to https://mcp-server-ip:9494/: TypeError: fetch
 failed
2026-07-31 15:23:36.944 [info] Stopping server bes-mcp-server
```

Symptom: Client fails to establish the connection or trust the listener certificate.

Solution: Inspect the listener certificate and ensure its Subject Alternative Name (SAN) exactly matches the configured MCP URL. Trust the certificate in the local or remote OS environment where the VS Code extension host is running, then reload the window. Alternatively you can use the http mode if https is not strictly required.

Note: This refers to an AI Client VSCode Output log error.

```
Fri, 31 Jul 2026 15:24:11
 +0200|INFO||MCP|TOOL|AUTH_FAILURE|10.14.85.112|Authentication failure
[auth_fingerprint v1:cb726c3527c3c35e error
"failed to create BigFix client: failed to read CA cert file from
C:/Program Files (x86)/BigFix Enterprise/BES MCP Server/data/ca2.crt:
open C:/Program Files (x86)/BigFix Enterprise/BES MCP Server/data/ca2.crt:
 The system cannot find the file specified."
```

Symptom: The client connects, but the MCP Server cannot validate the upstream Root Server TLS certificate or path.

Solution: Verify the path specified in bigfix.ca_cert_path within the config.yaml file. Ensure the file exists, contains the correct Root CA, and is readable by the unprivileged service account. Restart the service to apply changes.

Note: This refers to an audit log error.

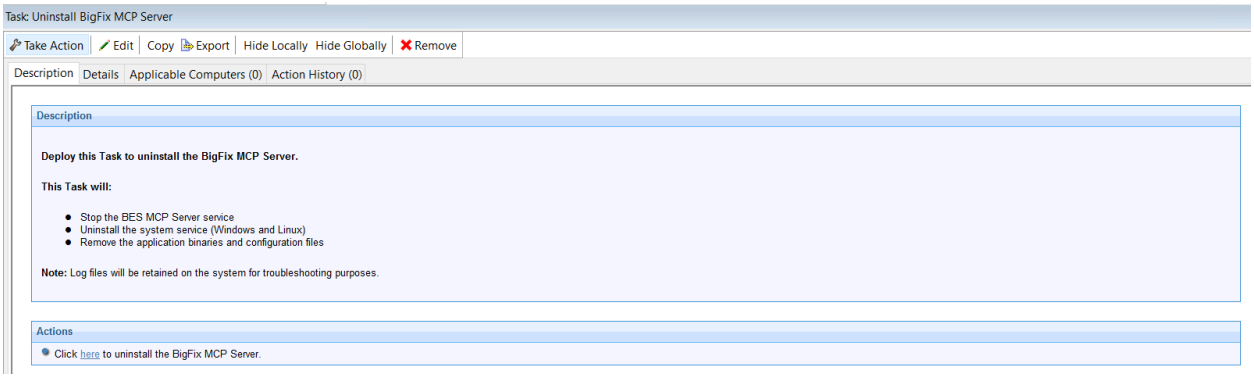
```
Fri, 31 Jul 2026 15:26:34 +0200 -- 6032 -- ListenAndServeTLS
[error="tls: found a certificate rather than a key in the PEM for the
 private key"]
```

Symptom: The MCP Server service fails to bind the port or load the listener TLS material.

Solution: Open config.yaml and verify the mcp_server.tls block. Ensure that key_path points to a valid private key file and the cert_path points to a valid certificate. Restart the bes-mcp-server service to apply the fix.

Note: This refers to an BESMCPServer.log error.

Chapter 8. Uninstalling BigFix Platform MCP Server



On Windows, the **Uninstall BigFix MCP Server** Fixlet stops the bes-mcp-server service and runs the executable service uninstall command. It then waits until the process fully terminates, moves any .log files from the data directory into the main installation directory, deletes the remaining data folder, and removes the MCP Server executable. The log files are intentionally preserved for troubleshooting purposes.

On Linux, the **Uninstall BigFix MCP Server** Fixlet stops the bes-mcp-server service and runs the executable service uninstall command from `/opt/BESMCPServer/bin`. It waits until the process terminates, moves any log files from `/var/opt/BESMCPServer/data` to `/var/log`, deletes the configuration and data directory, and finally removes the entire `/opt/BESMCPServer` installation tree. The service cleanup also handles the relevant systemd service files, while logs remain available outside the removed installation directory.

Appendix A. Support

For more information about this product, see the following resources:

- BigFix Support Portal
- BigFix Developer
- BigFix Playlist on YouTube
- BigFix Tech Advisors channel on YouTube
- BigFix Forum

Notices

This information was developed for products and services offered in the US.

HCL may not offer the products, services, or features discussed in this document in other countries. Consult your local HCL representative for information on the products and services currently available in your area. Any reference to an HCL product, program, or service is not intended to state or imply that only that HCL product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any HCL intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL product, program, or service.

HCL may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

For license inquiries regarding double-byte character set (DBCS) information, contact the HCL Intellectual Property Department in your country or send inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

HCL TECHNOLOGIES LTD. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. HCL may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-HCL websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this HCL product and use of those websites is at your own risk.

HCL may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by HCL under terms of the HCL Customer Agreement, HCL International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-HCL products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL has not tested those products and cannot confirm the accuracy of performance, compatibility or

any other claims related to non-HCL products. Questions on the capabilities of non-HCL products should be addressed to the suppliers of those products.

Statements regarding HCL's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to HCL, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. HCL, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS," without warranty of any kind. HCL shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:
© (your company name) (year).
Portions of this code are derived from HCL Ltd. Sample Programs.

Trademarks

HCL Technologies Ltd. and HCL Technologies Ltd. logo, and hcl.com are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other product and service names might be trademarks of HCL or other companies.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the HCL website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of HCL.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of HCL.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

HCL reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by HCL, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

HCL MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.