BigFix

MCP Server for Vulnerability Remediation

Special notice

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

Edition notice

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

Contents

Chapter 1. Introduction

The MCP Server for Vulnerability Remediation enables compatible AI clients to discover security patch information and generate BigFix Fixlet XML through the Model Context Protocol (MCP).

The server runs as a standalone service and exposes a streamable HTTP endpoint, normally over HTTPS on port `9495`. An MCP client supplies a BigFix REST API bearer token in the HTTP `Authorization` header. The server validates that token against the configured BigFix Server and uses it to preserve the identity and access controls associated with the caller.

The server provides a guided workflow that can:

- Look up patch data associated with a CVE.
- Discover downloadable patch files from supported vendor sources.
- Select an appropriate embedded BigFix template.
- Generate BES XML and a supporting result report.

Important: Review and test the generated content before you deploy it. The server does not create or deploy BigFix actions, modify managed endpoints, or bypass BigFix permissions.

How the server works

The server combines data from security and vendor services with embedded Fixlet templates. Vendor HTTPS connections are restricted by an embedded SHA3-256 Subject Public Key Information (SPKI) pin policy. The BigFix Server uses a separately configured CA certificate because that trust material is owned by the deploying organization.

After authentication and Remediate entitlement validation, the usual workflow is:

1. Retrieve CVE and patch data.
2. Retrieve patch file metadata for the selected vendor, product, and architecture.
3. Review file hashes, download locations, product selection, and warnings.
4. Generate and review the Fixlet.

For the complete workflow, see Using the server *(on page 14)*. For trust boundaries and security controls, see Security architecture *(on page 7)*.

Chapter 2. Supported scenarios

The MCP Server for Vulnerability Remediation supports CVE patch discovery, vendor patch file discovery, and Fixlet generation for a defined set of update flows.

The MCP Server for Vulnerability Remediation supports the following scenarios.

CVE patch discovery

Given a CVE identifier, the server retrieves available patch information and can enrich the result with product and CPE data. A CVE may map to multiple products or KB articles. You must select and verify the intended product before you continue.

Microsoft patch file discovery

The server searches the Microsoft Update Catalog by KB article or catalog query and returns matching patch file metadata. Supported target architectures are `x64`, `x86`, and `arm64`.

Microsoft and BigFix compatibility fields use the vendor-published SHA-1 and SHA-256 algorithms. These values are file integrity metadata required by BigFix `prefetch` commands; they are separate from the SHA3-256 algorithm used for certificate pinning.

Google Chrome patch discovery

The server retrieves Google Chrome release and enterprise installer information for a requested channel and architecture. Evergreen download URLs are not version-pinned. Retain any warning returned with the metadata and consider it during review.

WinGet package discovery

The server retrieves installer metadata from WinGet manifests for a package identifier, version, and architecture. A WinGet manifest hash indicates that the file metadata was published in the manifest; it does not replace deployment testing.

Fixlet generation

The server selects an embedded template and generates BES XML for supported flows, including:

- Windows cumulative updates.
- Windows checkpoint and cumulative updates.
- Microsoft KB executable updates.
- .NET Framework cumulative updates.
- MSI application updates.
- EXE application updates.

Review behavior and vendor warnings before you import or deploy the content.

Unsupported scenarios

The current server does not:

- Import generated Fixlets into a BigFix site.
- Create, stop, retry, or delete BigFix actions.
- Manage computers, operators, roles, sites, analyses, baselines, or tasks.
- Automatically deploy generated content.
- Replace product-owner validation of applicability or vendor authenticity.

Chapter 3. Security architecture

The MCP Server for Vulnerability Remediation is a standalone service with separate trust boundaries for MCP clients, the BigFix Server, and external patch intelligence providers.

The MCP client connects to the MCP Server for Vulnerability Remediation over HTTPS on port `9495` with a bearer token. The generator authenticates against the BigFix Server on port `52315`. Pinned HTTPS metadata from NVD, MSRC, the Microsoft Update Catalog, Google sources, and WinGet manifests flows to the generator; generated BES XML and reports return to the MCP client.

Request flow

1. An MCP client connects to the streamable HTTP endpoint, normally over HTTPS on port `9495`.
2. The client sends its BigFix REST API token in the `Authorization` header.
3. The server keeps the authorization value in the request context and validates it against the configured BigFix Server REST API, normally on port `52315`.
4. Immediately after successful authentication, the server uses the same bearer token to validate the user's Remediate entitlement against a vendor-defined policy embedded in the signed binary.
5. Only a successful entitlement response authorizes the request; all other responses are returned as a Remediate license denial.
6. The selected MCP tool validates its input.
7. The server contacts only the vendor hosts built into its pin policy.
8. The server formats patch metadata or generates Fixlet content and returns the result to the MCP client.

The server does not create a shared privileged BigFix identity. BigFix authentication remains authoritative for access to the MCP endpoint.

TLS trust directions

Two independent TLS relationships are configured differently:

Table 1. TLS trust directions

Connection	Trust mechanism
MCP client to MCP server	A configured listener certificate and private key, or a server-generated self-signed certificate. The MCP client must trust this certificate.
MCP server to BigFix Server	The CA certificate configured by `bigfix.ca_cert_path`. The server fails closed if the certificate cannot be read or validated.
MCP server to vendor services	A SHA3-256 SPKI pin policy embedded in the signed binary. Customers do not supply a runtime pin file.

Embedded vendor certificate pins

The server first performs normal X.509 certificate-chain and hostname validation. It then calculates a SHA3-256 digest over the SPKI of certificates in the verified chain and requires at least one configured pin to match.

Pin updates require a reviewed source change and a newly built and signed server binary. You cannot convert existing hash values into SHA3-256 pins; you must calculate pins from certificate SPKI data.

Authentication and token handling

- Tokens belong in the MCP client's `Authorization` header, not in `config.yaml`.
- The server validates the token against the configured BigFix Server.
- The authenticated user must also satisfy the embedded Remediate entitlement policy.
- Raw authorization tokens are not written to logs.

- Audit correlation uses a truncated, process-keyed HMAC-SHA3-256 fingerprint rather than the token value.
- Repeated authentication failures are subject to the configured IP lockout controls.

Tool execution and HITL

The three current tools perform lookup, metadata retrieval, and local content generation. They are registered as read operations and do not execute BigFix write operations. Consequently, the tools do not invoke the server's Human-in-the-Loop approval flow for BigFix writes. Generated content still requires human review before import or deployment.

For Microsoft KB Fixlets, `generate_fixlet` does not trust client-carried CVE or file metadata. It independently verifies the CVE-to-KB/product/architecture relationship through MSRC and requires every file field to match a fresh Microsoft Update Catalog lookup before rendering. This check fails closed when vendor verification is unavailable. Application generation uses trusted HTTPS source-host enforcement and hash validation; full package provenance binding remains dependent on adding the originating package identifier to that workflow.

Trusted payload hosts

The shipped payload-source allowlist is maintained in `config/trusted-download-hosts.json` and embedded into each server binary. Startup fails if the policy has an unsupported schema version or contains an empty, malformed, duplicate, non-normalized, or unsorted host list. Only exact lowercase DNS hostnames are accepted; wildcard domains, URLs, ports, and IP literals are prohibited.

`FIXLET_TRUSTED_DOWNLOAD_HOSTS` is an additive deployment override for publisher hosts that are not appropriate for every release. It cannot remove or replace shipped entries. Editing the embedded policy changes which origins may supply commands' payloads, so additions require the same security review and release rebuild as other bundled trust policy changes.

Logging and audit

The server maintains an operational log at the configured `log.file_path` and an audit log named `mcp_audit.log` in the workspace. Protect both logs using operating-system permissions and include them in the organization's retention and incident-response procedures.

Chapter 4. Requirements

Verify the following requirements before you install the MCP Server for Vulnerability Remediation.

Deployment requirements

Table 2. Requirements

Requirement	Details
Supported operating systems	A supported 64-bit Windows Server environment on AMD64.
BigFix connectivity	The MCP server host must reach the BigFix Server REST API over HTTPS, normally TCP port `52315`.
BigFix authentication	Each MCP client session requires a valid BigFix REST API bearer token.
Remediate entitlement	The authenticated BigFix user must have the Remediate entitlement required by the vendor-defined policy embedded in the server binary.
MCP client connectivity	MCP clients must reach the server's streamable HTTP listener, normally TCP port `9495`.
MCP client capability	A client that supports remote streamable HTTP MCP servers and custom request headers.
BigFix TLS trust	A PEM-encoded CA certificate that validates the BigFix Server certificate and is readable by the MCP service account.
MCP listener trust	The MCP client must trust the certificate presented by the MCP server. The client URL must match a certificate SAN.
Vendor connectivity	Outbound HTTPS access to the vendor hosts required for the selected workflow, including Microsoft, Google, GitHub, NVD, and MSRC services.
Storage	Space for the binary, configuration, generated TLS material, logs, and response data. Store exported Fixlets in an organization-approved content repository.
Service account	An account with permission to run a Windows service and read/write the configured workspace.

Network summary

Table 3. Network summary

Source	Destination	Default port	Purpose
MCP client	MCP Server for Vulnerability Remediation	`9495`	Streamable HTTP MCP traffic, normally protected by TLS.
MCP Server for Vulnerability Remediation	BigFix Server	`52315`	Bearer-token validation through the BigFix REST API.
MCP Server for Vulnerability Remediation	Approved vendor services	`443`	CVE, patch, release, catalog, manifest, and download metadata.

Service account permissions

The Windows service account must be able to:

- Read and execute the server binary.
- Read `config.yaml` and the configured BigFix CA certificate.
- Read the listener private key when custom TLS material is used.
- Read and write the workspace and configured log locations.

- Bind the configured listener port.
- Establish outbound HTTPS connections to BigFix and supported vendor services.

A custom account must have the **Log on as a service** right. Grant it access to only the required installation, workspace, and log locations.

Chapter 5. Installing the server

Use the Windows AMD64 release package to install the MCP Server for Vulnerability Remediation as a Windows service.

Keep the binary, configuration, CA certificate, logs, and any client-exported output under locations protected by operating-system permissions.

A release package contains:

- The `bes-mcp-fixlet-gen` executable, with an `.exe` suffix on Windows.
- `data/config.example.yaml`.
- Supporting deployment documentation.

Create `data/config.yaml` from the example before you start the server. Do not place a BigFix token in this file.

To install the server on Windows, complete the following steps:

1. Extract the Windows package into a protected installation directory, for example `C:\Program Files\BES MCP Fixlet Generator`.
2. Copy `data\config.example.yaml` to `data\config.yaml`.
3. Copy the organization-approved BigFix CA certificate into the data directory or another location readable by the service account.
4. Edit `data\config.yaml` as described in Server configuration *(on page 11)*.
5. Open an Administrator PowerShell session in the installation directory.
6. Install and start the Windows service.

```
.\bes-mcp-fixlet-gen.exe service install --workspace "C:\Program Files\BES MCP Fixlet Generator\data"
.\bes-mcp-fixlet-gen.exe service start
```

7. **Optional:** To run under a dedicated service account, install the service with the account credentials.

```
.\bes-mcp-fixlet-gen.exe service install `
  --workspace "C:\Program Files\BES MCP Fixlet Generator\data" `
  --user "DOMAIN\service-user" `
  --password "<password>"
.\bes-mcp-fixlet-gen.exe service start
```

The service runs from the configured installation and workspace paths.

Confirm the binary identity:

```
bes-mcp-fixlet-gen version
```

Confirm that TCP port `9495` is listening and connect with a client that trusts the listener certificate. A plain HTTP `GET` can return an MCP protocol error; successful TLS connection without connection refusal is sufficient for this transport check.

Server configuration

Configure the listener, logging, and BigFix connection settings in `config.yaml`.

Example configuration

The following configuration uses the production BigFix REST API port `52315`:

```
mcp_server:
  scheme: "https"
  port: "9495"
  read_only: false
  disable_hitl: false
  max_auth_failures: 5
  lockout_duration_seconds: 300
  max_response_bytes: 1048576
  max_bulk_items: 100
```

```
  tls:
    dns_names:
      - "mcp-fixlet-gen.example.com"
      - "localhost"
    ip_addresses:
      - "127.0.0.1"
log:
  file_path: "logs/server.log"
  max_size_mb: 100
  max_backups: 3
  max_age_days: 28
  compress: true
bigfix:
  url: "https://bigfix.example.com:52315"
  ca_cert_path: "bigfix-ca.pem"
  auth_timeout_seconds: 30
```

Relative paths are resolved from the workspace supplied to `service install`. The service fails to start if `bigfix.ca_cert_path` is absent, unreadable, or does not contain valid PEM certificate material.

Listener settings

Table 4. Listener settings

Setting	Description
`mcp_server.scheme`	Use `https` for deployed environments. Use `http` only for isolated local testing.
`mcp_server.port`	MCP listener port. The documented default for this server is `9495`.
`mcp_server.tls.dns_names`	DNS names included in an automatically generated listener certificate.
`mcp_server.tls.ip_addresses`	IP addresses included in an automatically generated listener certificate.
`mcp_server.tls.cert_path`	Optional custom listener certificate path. Configure together with `key_path`.
`mcp_server.tls.key_path`	Optional private key matching `cert_path`.

The hostname or IP in the MCP client URL must appear in the certificate Subject Alternative Name (SAN).

Runtime security settings

Table 5. Runtime security settings

Setting	Description
`mcp_server.max_auth_failures`	Consecutive authentication failures allowed before source-IP lockout.
`mcp_server.lockout_duration_seconds`	Duration of an authentication lockout.
`mcp_server.max_response_bytes`	Maximum accepted downstream response size.
`mcp_server.max_bulk_items`	Maximum accepted bulk item count.

The current Fixlet Generator tools do not execute BigFix write operations. The generic `read_only` and `disable_hitl` settings remain part of the shared server runtime but do not replace review of generated Fixlets.

BigFix settings

Table 6. BigFix settings

Setting	Description
`bigfix.url`	BigFix Server REST API base URL, normally `https://<bigfix-server>:52315`.
`bigfix.ca_cert_path`	PEM certificate or CA bundle used to validate the BigFix Server certificate.
`bigfix.auth_timeout_seconds`	Timeout for bearer-token authentication and entitlement validation requests.

After bearer-token authentication succeeds, the server validates the user's Remediate entitlement using vendor-defined policy embedded in the signed binary. The entitlement policy is not customer-configurable.

Restarting after configuration changes

Restart the service whenever `config.yaml`, the listener TLS material, or the BigFix CA certificate changes.

```
.\bes-mcp-fixlet-gen.exe service restart
```

Chapter 6. Using the server

The MCP Server for Vulnerability Remediation exposes a streamable HTTP MCP endpoint and requires a BigFix REST API bearer token on every client request.

Configure the MCP client, run the standard workflow, and review the generated content before you import or deploy it.

Configuring Visual Studio Code

Configure Visual Studio Code to connect to the MCP Server for Vulnerability Remediation over the streamable HTTP MCP endpoint.

1. Add a `.vscode/mcp.json` file to the client workspace.

```
{
  "servers": {
    "bes-mcp-fixlet-gen": {
      "type": "http",
      "url": "https://mcp-fixlet-gen.example.com:9495/",
      "headers": {
        "Authorization": "Bearer ${input:bigfixToken}"
      }
    }
  },
  "inputs": [
    {
      "id": "bigfixToken",
      "type": "promptString",
      "description": "BigFix REST API token",
      "password": true
    }
  ]
}
```

2. Run **MCP: List Servers** from the Command Palette.
3. Start or restart `bes-mcp-fixlet-gen`.
4. Supply the BigFix token when prompted.
5. Open Chat in Agent mode and confirm that three tools are available.

Important: Do not store the token directly in `mcp.json` or `config.yaml`.

Configuring Claude Desktop

Configure Claude Desktop to connect to the MCP Server for Vulnerability Remediation through an HTTP-to-stdio bridge.

Claude Desktop commonly launches local MCP servers through stdio. Because the MCP Server for Vulnerability Remediation exposes a streamable HTTP endpoint, configure an HTTP-to-stdio bridge such as `mcp-remote`.

1. Edit the Claude Desktop configuration file.

   macOS: `~/Library/Application Support/Claude/claude_desktop_config.json`

```
{
  "mcpServers": {
    "bes-mcp-fixlet-gen": {
      "command": "npx",
      "args": [
        "-y",
        "mcp-remote",
        "https://127.0.0.1:9495/",
        "--header",
        "Authorization:${AUTH_HEADER}"
      ],
      "env": {
```

```
        "AUTH_HEADER": "Bearer <bigfix-token>"
      }
    }
  }
}
```

2. Restart Claude Desktop completely.

   For a shared server, replace the URL with its DNS name or IP address. Keep the space after `Bearer` in the environment value. Node-based clients may also require `NODE_EXTRA_CA_CERTS` to point to the copied `selfsigned_cert.pem` certificate. Completely restart Claude Desktop after changing the configuration.

Important: Claude Desktop does not support the VS Code-style interactive token input. This example stores the token as plain text in `claude_desktop_config.json`; restrict access to the file and use a dedicated, revocable token. Where secrets in configuration files are prohibited, use a wrapper that retrieves the token from the operating-system credential store and starts `mcp-remote` with `AUTH_HEADER` set.

Configuring ChatGPT Desktop

Configure ChatGPT Desktop to connect to the BES MCP Fixlet Generator when the client supports remote streamable HTTP MCP servers.

ChatGPT Desktop MCP support depends on the installed release and organization policy.

1. Add the server endpoint through the client's MCP or connector settings.

   ```
   https://127.0.0.1:9495/
   ```

2. Configure the connection to send the authorization header on every request.

   ```
   Authorization: Bearer <bigfix-token>
   ```

3. **Optional:** If the client does not support remote streamable HTTP MCP servers directly, use an HTTP-to-stdio bridge. Use an HTTP-to-stdio bridge with the same URL and authorization-header pattern shown for Claude Desktop. If the client exposes neither remote HTTP nor stdio MCP configuration, use VS Code, Claude Desktop, or another compatible MCP client.

Standard workflow

Use the standard workflow to retrieve CVE patch data, retrieve patch files, and generate a Fixlet.

1. Retrieve CVE patch data.

   Example request:

   ```
   {
     "cve_id": "CVE-2025-21179"
   }
   ```

   Review the returned KB articles, products, CPE data, patch availability, and any selection warning. Do not continue until the intended product is unambiguous.
2. Retrieve patch files.

   Microsoft example:

   ```
   {
     "vendor": "microsoft",
     "kb_article_id": "5051987",
     "architecture": "x64"
   }
   ```

   WinGet example:

```
{
  "vendor": "winget",
  "package_id": "Mozilla.Firefox",
  "version": "145.0",
  "architecture": "x64"
}
```

Review every filename, URL, size, hash, and verification warning. Vendor metadata is an input to generation, not deployment approval.

3. Generate the Fixlet.

   Pass the selected product and patch files to `generate_fixlet`. A simplified Microsoft example is:

```
{
  "cve_id": "CVE-2025-21179",
  "kb_article_id": "5051987",
  "product_name": "Windows 11 Version 24H2 for x64-based Systems",
  "severity": "Important",
  "files": [
    {
      "filename": "windows11-kb5051987-x64.msu",
      "sha1": "<vendor-sha1>",
      "sha256": "<vendor-sha256>",
      "size_bytes": 123456,
      "download_url": "https://catalog.sf.dl.delivery.mp.microsoft.com/<path>"
    }
  ]
}
```

   Use the exact metadata returned by the lookup tools. Do not substitute placeholder hashes or file sizes in production content.

The generator accepts only HTTPS downloads from trusted vendor hosts. The release-shipped policy is embedded in the binary from `config/trusted-download-hosts.json`; it includes the Microsoft Update Catalog and Google Chrome hosts. Maintainers add broadly approved vendor hosts to that source-controlled file, keep entries lowercase and lexicographically sorted, review the change as security policy, and rebuild the binary.

WinGet installer hosts vary by publisher. An administrator can append deployment-specific exact hostnames at startup without changing the embedded policy:

```
export FIXLET_TRUSTED_DOWNLOAD_HOSTS="download.mozilla.org,downloads.vendor.example"
```

The setting is a comma-separated list of exact DNS hostnames. Wildcards, URL paths, IP addresses, HTTP URLs, user information, and nonstandard ports are not accepted. A URL supplied to `generate_fixlet` is rejected unless its normalized hostname is in the embedded or administrator-configured allowlist. Invalid embedded policy prevents server startup; invalid runtime entries are rejected during URL validation.

Review the generated content before you import or deploy it. For details, see Reviewing generated content *(on page 16)*.

Reviewing generated content

Review a generated Fixlet before you import or deploy it.

Before you import or deploy a generated Fixlet, review:

- The selected product, architecture, CVE, KB article, and fixed version.
- Relevance expressions and applicability boundaries.
- Download URL, filename, size, SHA-1, and SHA-256 metadata.
- Installation commands, arguments, exit-code handling, and restart behavior.
- Prerequisite and checkpoint sequencing.
- Vendor warnings and verification statements.

The server generates content but does not import, approve, or deploy it.

For exact inputs, see the Tool reference *(on page 18)*.

Chapter 7. Tool reference

The server exposes three MCP tools. The input schemas are discoverable through MCP and are summarized here for operational review.

get_cve_patch_data

Retrieves patch and optional CPE enrichment for a CVE.

Table 7. get_cve_patch_data inputs

Input	Required	Description
`cve_id`	Yes	CVE identifier, for example `CVE-2025-21179`.

The result can contain multiple KB or product choices. Observe selection warnings and pass the selected values to later tools.

fetch_patch_files

Retrieves patch file metadata from a supported vendor.

Table 8. fetch_patch_files inputs

Input	Required	Description
`vendor`	No	`microsoft`, `google_chrome`, or `winget`. Defaults to `microsoft`.
`kb_article_id`	Conditional	Microsoft KB number without the KB prefix. Use this or `catalog_query` for Microsoft.
`catalog_query`	Conditional	Microsoft Update Catalog search query. Use this or `kb_article_id` for Microsoft.
`architecture`	No	`x64`, `x86`, or `arm64`. Defaults to `x64`.
`channel`	Conditional	Google Chrome release channel. Defaults to `stable` where applicable.
`package_id`	Conditional	WinGet package identifier.
`version`	Conditional	Application version used by WinGet or application update flows.

The result includes file metadata and verification or provenance warnings. Retain those warnings through review.

generate_fixlet

Selects an embedded template and generates BES XML.

Table 9. generate_fixlet inputs

Input	Required	Description
`cve_id`	Yes	CVE identifier used in Fixlet metadata.
`kb_article_id`	Conditional	Microsoft KB number without the prefix. Provide this or `fixed_version`.
`fixed_version`	Conditional	Fixed application version for non-KB application updates. Provide this or `kb_article_id`.
`product_name`	Yes	Selected product name.
`severity`	No	Security severity when available.
`release_date`	No	Patch release date when available.

Table 9. generate_fixlet inputs (continued)

Input	Required	Description
`fixed_build_number`	No	Fixed OS or product build number when available.
`publisher`	Conditional	Application publisher for application update flows.
`display_name_prefix`	Conditional	Installed application display-name prefix.
`cpe_id`	No	CPE 2.3 identifier when available.
`files`	Yes	One or more patch file objects returned by `fetch_patch_files`.

Each `files` item supports:

Table 10. files item fields

Field	Required	Description
`filename`	Yes	Patch filename.
`sha1`	Conditional	SHA-1 file hash. Required for KB/MSU generation and BigFix compatibility.
`sha256`	No	SHA-256 file hash when available.
`size_bytes`	Yes	Exact file size in bytes.
`download_url`	Yes	Vendor download URL.

Note: SHA-1 and SHA-256 in this schema are vendor file-integrity fields used by BigFix `prefetch`. They are not certificate-pin algorithms. Vendor TLS certificate pins use SHA3-256.

For Microsoft KB generation, the server independently rechecks provenance before rendering. Every CVE must map through MSRC to the submitted KB, product, and architecture, and every submitted file must exactly match the current Microsoft Update Catalog filename, SHA-1, SHA-256, size, URL, count, and order. Generation fails closed if either verification source is unavailable or any value differs.

Application update flows currently enforce trusted HTTPS source hosts and prefetch integrity fields, but do not have the same end-to-end vendor provenance binding because `generate_fixlet` does not yet receive the originating package ID or channel. Preserve the `fetch_patch_files` result and its verification warnings during application review.

Tool failure behavior

Tools fail closed for invalid input, unavailable vendor data, unsupported combinations, or generation validation errors. Treat warnings in successful results as part of the result; success does not mean that generated content is approved for deployment.

Chapter 8. Troubleshooting

Use the operational log configured by `log.file_path`, the workspace `mcp_audit.log`, and the MCP client's output log when you diagnose a problem.

Important: Never publish raw bearer tokens or private keys with diagnostic data.

Server fails to start because the BigFix CA is missing

Example symptom:

```
failed to read BigFix CA certificate: open <path>: no such file or directory
```

Verify that `bigfix.ca_cert_path` resolves from the configured workspace, contains the approved PEM certificate or CA bundle, and is readable by the service account. Restart the service after you correct the path or permissions.

MCP client reports TypeError: fetch failed

Common causes are:

- The service is not listening on port `9495`.
- The listener certificate is not trusted by the environment running the MCP client.
- The client URL hostname or IP does not match a certificate SAN.
- A firewall blocks client-to-server traffic.

Check the listener, verify the certificate SAN, install trust in the environment running the client, and restart the client. For remote development environments, trust must exist where the MCP extension host runs.

Authentication fails

Verify that:

- The client sends the complete `Authorization: Bearer <token>` header.
- The token is valid for the configured BigFix Server.
- `bigfix.url` uses HTTPS and production port `52315`.
- The MCP host can reach the BigFix REST API.
- The configured CA validates the certificate presented by BigFix.

Repeated authentication failures can lock the source IP for `lockout_duration_seconds`. Wait for the lockout to expire after you correct the credentials or connection.

Remediate entitlement denial

If authentication succeeds but the user does not satisfy the embedded entitlement policy, the server returns `403 Forbidden` and reports that the tool is available only with a Remediate license.

Verify that:

- The authenticated BigFix user has an active Remediate entitlement.
- The client is connected to the intended BigFix Server configured by `bigfix.url`.
- The bearer token belongs to the intended user and has not expired.

An entitlement denial does not count as a failed credential attempt and does not trigger the authentication lockout counter.

Tools are missing

Run **MCP: List Servers** and inspect the server output. Confirm that the server is connected and the BigFix token was supplied. Restart the MCP client entry after you change its configuration.

The expected tool names are:

```
get_cve_patch_data
fetch_patch_files
generate_fixlet
```

Vendor certificate pin mismatch

A pin mismatch causes the vendor connection to fail closed. Confirm system time and ordinary CA trust first. Do not disable TLS validation or replace the embedded policy at runtime.

Vendor certificate rotations require a newly built and signed server binary containing reviewed SHA3-256 SPKI pins. Install the approved update or escalate the failure to the product maintainer with the hostname, timestamp, server version, and redacted log entry.

Patch lookup returns multiple products or files

This can be a valid vendor response. Refine the CVE, KB, catalog query, architecture, channel, package identifier, or version. Review product names and classifications rather than selecting the first result automatically.

Fixlet generation rejects missing SHA-1

Microsoft KB/MSU flows require SHA-1 because BigFix `prefetch` compatibility uses both vendor SHA-1 and SHA-256 metadata. Retrieve the file through `fetch_patch_files` and pass its exact metadata. Do not calculate SHA3-256 and place it in a `sha1` or `sha256` field.

Generated XML is not written to disk

The server returns generated XML in the MCP tool response; it does not write a Fixlet file to a configured output directory. Save the reviewed response through the MCP client or an approved content workflow. If that save fails, check the client user's permissions and available disk space.

Service commands fail

Run installation and removal commands from an Administrator PowerShell session. Keep the same `--workspace` location used at installation when you reinstall or diagnose service configuration.

Chapter 9. Uninstalling the server

Uninstalling the service does not replace data-retention decisions. Back up required logs and any Fixlets exported through the client workflow before you delete the installation or workspace.

Before you uninstall:

1. Record the installed server version.
2. Stop MCP client connections.
3. Back up `mcp_audit.log`, the configured operational logs, and client-exported Fixlets according to organizational policy.
4. Record the installation and workspace paths.

Important: Do not include bearer tokens or listener private keys in diagnostic archives unless an approved security process requires them.

To uninstall the server on Windows, complete the following steps:

1. Open an Administrator PowerShell session in the directory containing the executable.
2. Stop and uninstall the Windows service.

```
.\bes-mcp-fixlet-gen.exe service stop
.\bes-mcp-fixlet-gen.exe service uninstall
```

3. After you confirm that the Windows service no longer exists, remove the installation and workspace directories according to the organization's retention policy.
4. **Optional:** Remove listener certificate trust from client systems only if no other deployment uses that certificate or CA.
5. **Optional:** If a dedicated service account is no longer needed, remove it through the organization's identity-management process after you confirm that it owns no retained files or other services.

After you uninstall the server:

- Confirm that no process is listening on port `9495`.
- Confirm that the service no longer appears in Windows Services.
- Confirm that retained logs are protected and readable only by authorized users.
- Remove stale MCP client entries that point to the uninstalled server.

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.