BigFix Platform 11.0.6

Administrator & Relevance Quick-Reference Guide

A practical, independently written reference covering platform architecture, console administration, content types, the Relevance language, and common inspector patterns for BigFix Platform 11.0.x.

Compiled July 2026

Contents

1. Platform Architecture

BigFix Platform 11.0.6 is built around a small set of components that work together to give near real-time visibility and control over managed endpoints. Understanding how they fit together is the foundation for every other administrative task.

1.1 Core Components

Component	Role
Root Server	Coordinates content distribution, stores state in the BigFix SQL database, and is the authority for the deployment. Supports Disaster Server Architecture (DSA) for redundancy — a warm-standby secondary server that mirrors the primary.
Relay	A client promoted to also cache and forward content and gathers, reducing WAN load and server connection counts. Relays can chain to other relays, forming a hierarchy.
Client (Agent)	Runs on every managed endpoint. Evaluates Relevance locally, reports results upward, and executes Action Script when a relevant Fixlet/Task/Baseline is deployed. Designed to be lightweight (well under 10 MB RAM in typical use).
Console	The primary administrative interface (Windows-based, or the newer web console) used by Master Operators and Console Operators to author and deploy content.
WebUI	A browser-based interface, generally used for day-to-day operational tasks (patching, queries, dashboards) by a broader set of operators without requiring the full Console.
Web Reports	A reporting front end that aggregates data — optionally from multiple root servers — for dashboards, audit trails, and data export.
SQL Database	Stores server-side state: computer data, content, actions, and results. Microsoft SQL Server is the supported database engine.

1.2 Typical Topology

A standard deployment is hierarchical: the Root Server gathers content (Fixlets, patches, updates) from external and internal sites, relays fan that content out to reduce direct server connections, and clients pull from their assigned relay. Client-to-relay assignment is normally automatic (based on relay affiliation/locate-server logic) but can be manually pinned. This structure scales to very large estates without requiring every endpoint to talk directly to the server.

```
Root Server
  |-- Relay (Region A)
  |     |-- Client, Client, Client ...
  |-- Relay (Region B)
  |     |-- Relay (child)
  |     |     |-- Client, Client ...
  |     |-- Client, Client ...
```

```
|-- (Optional) DSA Warm-Standby Root Server
```

1.3 Content Sites

All Fixlets, Tasks, Baselines, and Analyses live inside Sites — logical containers that group related content and control who can see and act on it. Sites are typically one of:

- **External sites** — vendor-maintained content such as Patches for Windows, Patches for RHEL, or application-specific update sites, refreshed automatically via the gather process.
- **Operator sites** — created by a Master Operator for custom or internal content, with subscription and authoring rights assigned per operator.
- **Action site / Master Operator site** — the top-level administrative site controlling operator accounts, roles, and global settings.

2. Console Administration & Security Model

2.1 Operator Roles

Role	Typical Scope
Master Operator (MO)	Full administrative rights: creates/removes Console Operators, manages all sites, unrestricted computer scope, can assign roles.
Console Operator (Non-Master)	Rights limited to specific sites and/or a specific set of computers (management rights). Cannot create other operators.
WebUI Operator	Uses the WebUI rather than the full Console; permission sets there are managed independently and are typically narrower and task-focused.
LDAP / Directory-Integrated Operator	Authenticates against an external directory (LDAP/Active Directory) rather than a local BigFix password, while still holding a BigFix role and scope.

2.2 Management Rights & Computer Scope

Non-master operators are restricted to a defined computer scope, set either as an explicit list or — more commonly at scale — as a Relevance-based computer assignment. This lets an operator's authority automatically track a changing population (e.g., "all computers whose name starts with 'FIN-'") without manual list maintenance.

2.3 Roles vs. Permissions

Two layers control what an operator can do: **site membership** (which content sites the operator can see and author in) and **permissions/roles** (finer-grained rights such as custom content authoring, the ability to send refresh/wake signals, or console-vs-WebUI access). Master Operators assign both when creating or editing an operator account.

2.4 Common Console Administration Tasks

- Creating and scoping a new Console Operator, then subscribing it to the required sites.
- Enabling or disabling external sites (e.g., Patches for Windows) and setting computer subscriptions per site.
- Building manual and automatic Computer Groups to organize targeting.
- Reviewing the Audit Log (BigFix Administration Tool) for who took what action and when.
- Managing Action signing — operators typically sign actions with a private key/password so that clients can cryptographically verify content authenticity before executing it.

3. Content Types

Every piece of work BigFix does is expressed through a small number of content types. They share the same Relevance-driven applicability model but differ in lifecycle and intent.

3.1 Fixlet

A Fixlet pairs a Relevance clause (the applicability test) with one or more Action scripts (the remediation). A Fixlet is "relevant" on a client exactly when its Relevance evaluates true — meaning the condition it detects (missing patch, misconfiguration, absent software) still exists. Once remediated, the Fixlet naturally becomes not-relevant and drops off the client's relevant list; if the condition recurs, it becomes relevant again automatically.

3.2 Task

A Task looks like a Fixlet but represents a one-time or recurring operational action (e.g., restart a service, run a maintenance script) rather than a compliance/vulnerability state. After a Task's actions complete, it is marked not-relevant on that client and — unlike a Fixlet — its Relevance is not re-evaluated to confirm the outcome; Success Criteria are used instead to judge whether the run succeeded.

3.3 Baseline

A Baseline bundles multiple Fixlets and/or Tasks into a single deployable unit with a defined execution order, useful for multi-step rollouts (install → patch → configure). A Baseline's own Applicable Computers criteria are combined (ANDed) with each component's own Relevance.

3.4 Analysis

An Analysis is a set of Relevance-based property expressions used to collect and report data from clients (inventory, configuration state, custom facts) rather than to remediate anything. Analyses back most dashboards and drive computer properties used for targeting.

3.5 Action / Action Script

Action Script is the imperative half of a Fixlet or Task — the actual commands run on the endpoint. Relevance expressions can be embedded inside Action Script using curly braces `{ }`, letting a single script adapt itself per client (e.g., resolving an install path that differs machine to machine).

3.6 Computer Groups

Type	Behavior
Manual Group	Static membership, explicitly selected by an operator.
Automatic Group	Membership is a live Relevance query; computers move in and out automatically as their state changes.

4. Relevance Language Fundamentals

Relevance is BigFix's read-only query language. Every expression asks a question about the state of a client (or, in Session Relevance, the server database) and returns an answer — or an error if the question doesn't apply. Content authoring is mostly the skill of writing correct, efficient Relevance.

4.1 Core Concepts

- **Inspectors** — built-in functions that expose data about the system (files, registry, processes, services, installed software, hardware, and hundreds of other facets). An inspector call reads like natural English, e.g. `name of operating system`.
- **Properties** — every inspector returns an object with its own properties, which can themselves be queried; this chains naturally, e.g. `version of file "notepad.exe" of folder "windows" of system folder`.
- **Plurals** — most singular inspectors have a plural form returning a set instead of a single value (and erroring if none exist), which avoids "singular expression refers to nonexistent object" errors when zero or multiple matches are possible.
- **Types & casts** — values have types (string, integer, boolean, file, etc.) and can be cast, e.g. `(value of it as string)`.
- **Client vs. Session Relevance** — Client Relevance runs on the endpoint against local state; Session Relevance runs against the BigFix relay/server database (used in Web Reports, BigFix Query, and dashboards) and shares the same core syntax.

4.2 Syntax Building Blocks

Construct	Purpose	Example
exists	True/false test for whether an object or property is present, without raising an error if it is absent.	exists key "HKEY_LOCAL_MACHINE\Software\Foo" of registry
whose	Filters a set down to members matching a condition.	services whose (name of it = "Spooler")
if / then / else	Conditional expressions.	if exists file "x.txt" of folder "c:\" then "found" else "missing"
as	Casts a value to another type, e.g. lowercasing a string for case-insensitive comparison.	(name of operating system) as lowercase
of	Reads a property "of" an object; also used for scoping/optimizing evaluation order.	version of file "app.exe" of folder "c:\Program Files\App"
AND / OR	Boolean combination of relevance clauses.	(exists key ...) AND (version of it >= "2.0")

Construct	Purpose	Example
contains / matches	Substring or regex-style comparison on strings.	(name of it as lowercase) contains "chrome"

4.3 A Minimal Worked Example

A Fixlet applicability that flags Windows systems missing a specific registry-based marker:

```
exists true whose (
    (name of operating system as lowercase) contains "windows"
    AND NOT exists key
        "HKEY_LOCAL_MACHINE\SOFTWARE\Contoso\Agent\Version"
        of native registry
)
```

This is deliberately verbose to show the pattern: guard the OS family first, then test the condition you actually care about, combined with AND/OR/NOT as needed.

5. Inspector Categories & Practical Patterns

BigFix ships thousands of inspectors — far too many to enumerate meaningfully here (the full, searchable library lives on the official Relevance reference; see Appendix B). What's more useful day to day is knowing the major categories and having reliable, reusable patterns for the checks that come up constantly. This chapter is organized that way.

5.1 Major Inspector Categories

Category	What it covers
Operating system / platform	OS name, version, build, service pack, architecture, locale, boot time.
File system	Files, folders, existence, size, dates, versions, hashes, permissions.
Windows registry	Keys, values, existence and content checks under HKLM/HKCU/etc.
Processes & services	Running processes, installed services and their state (running/stopped/startup type).
Installed software	Add/Remove Programs-equivalent data, application inventory, package managers on Linux (rpm, dpkg).
Networking	IP configuration, DNS, connected state, MAC address, open ports.
Hardware	CPU, memory, disk, BIOS/firmware, manufacturer and model.
Users & security	Logged-on user, local accounts/groups, security policy settings.
BigFix client/agent state	Client version, relay affiliation, last report time, site subscriptions.
Date and time	Current time, uptime, scheduling-related comparisons.

5.2 Reusable Patterns

Check installed software (Windows) by name

```
exists (regapp "notepad.exe") OR
exists application whose (name of it as lowercase
    contains "adobe acrobat")
```

Compare a version number

```
exists file "app.exe" whose (
    (version of it as version) < (version "2.5.0" as version)
) of folder "c:\Program Files\App"
```

Casting to `version` rather than comparing raw strings avoids the classic "10.0" < "9.5" lexical-comparison trap.

Windows patch / hotfix presence

```
not exists hotfix "KB5034441"
```

Linux package check (RPM-based)

```
exists package "openssl" of rpm
exists rpm whose (name of it = "openssl" AND
    (version of it as string) < "3.0.7") of rpms
```

Service running state

```
exists service "Spooler" whose (status of it = "Running")
```

Registry key/value existence and content

```
exists key "HKEY_LOCAL_MACHINE\SOFTWARE\Contoso" of native registry
value "Enabled" of key "HKEY_LOCAL_MACHINE\SOFTWARE\Contoso"
    of native registry as string = "1"
```

Combine OS family + version guard (a very common shape)

```
(name of operating system as lowercase) contains "windows" AND
(version of operating system >= "10.0")
```

Date/time-based targeting

```
(now - (creation time of it)) > (86400 * 30) of file
    "install.log" of folder "c:\temp"
```

Useful for flagging stale files/logs, or gating maintenance windows.

Client/agent self-checks

```
version of client
name of site of bes relay
```

5.3 Debugging Relevance

- Use the **QnA tool** (bundled with the client) or the online evaluator on the developer site to test expressions interactively before putting them in a Fixlet.
- Wrap uncertain object references in `exists` before dereferencing properties, to avoid "nonexistent object" errors.
- Prefer plural forms (`files`, `services`) over singular when more than one match is possible.
- Push filtering into the `whose` clause and use `of` to scope early — this reduces the amount of data the client has to walk and materially improves evaluation performance at scale.

6. Action Script Essentials

Action Script is the command layer executed on a client when an operator takes an action on a relevant Fixlet, Task, or Baseline. It supports a set of built-in commands plus embedded Relevance substitution.

6.1 Common Commands

Command	Purpose
wait / waithidden	Run an executable and wait for completion (optionally hidden).
copy / move / delete	File operations, with sources resolvable via URL or relative download path.
createfile / appendfile / delete	Build small files inline in the script.
regset / regdelete	Registry modification commands (Windows).
continue if / waitcontinue if	Guard a step so the script halts (rather than errors) if a condition isn't met.
action requires restart	Signals that a reboot is needed for the change to take effect.

6.2 Embedding Relevance in Action Script

```
waithidden {pathname of regapp "notepad.exe"} /A
continue if {exists file "app.exe" of folder "c:\Program Files\App"}
```

6.3 Success Criteria

Every action can define Success Criteria — separate from the Fixlet's own Relevance — used to judge whether the run actually succeeded on that client (e.g., re-checking the Relevance, or checking the process exit code). This is especially important for Tasks, where the triggering Relevance is not re-evaluated automatically after the run.

7. WebUI Administration

The WebUI is a browser-based interface layered on top of the same Root Server, intended for day-to-day operational work — running patches, viewing dashboards, running BigFix Query — without requiring the full desktop Console.

7.1 Installation Model

WebUI is installed as an add-on against an existing BigFix Platform V11 (or later) deployment. It has its own service/process separate from the Root Server and typically sits behind the same or a dedicated HTTPS endpoint.

7.2 Permissions

WebUI access and function-level rights (e.g., which apps/dashboards an operator can open) are controlled independently from Console permissions, using WebUI-specific permission settings layered on top of the operator's existing BigFix role and computer scope.

7.3 BigFix Query via WebUI

For deployments with a Lifecycle or Compliance license (9.5.2+), the WebUI exposes BigFix Query — the ability to run ad hoc Client Relevance queries directly against live target computers and get results back interactively, which is often faster for one-off investigation than authoring a full Analysis.

8. Patch & Content Deployment Workflow

A representative end-to-end flow for rolling out a patch:

- **Enable** the relevant external site (e.g., Patches for Windows) and accept its license.
- **Subscribe** the site to the appropriate computer scope.
- **Gather** — either wait for the scheduled gather or trigger manually — to pull the site's Fixlets down to the server.
- **Review Relevant Fixlets and Tasks** for the target computer group to see what's currently applicable.
- **Take Action** on the Fixlet: choose target computers/groups, execution behavior (immediately / on a schedule / offer), and constraints (retry, deadline, user messaging).
- **Monitor** the Action's status pane as it moves through Not Evaluated → Evaluating → Running → Fixed/Failed per client.
- **Confirm** — once remediated, the Fixlet naturally drops off the relevant list for that client as its Relevance re-evaluates to false.

8.1 Targeting Options

Method	Use case
By retrieved properties	Target computers matching a live Relevance-based filter in the Take Action dialog.
By computer group	Target a pre-built manual or automatic group.
By explicit selection	Hand-pick specific computers from a list.

9. Maintenance, Troubleshooting & Best Practices

9.1 Health Checks

- Watch client report timestamps for stale/non-reporting endpoints — often a relay connectivity or certificate issue.
- Monitor Root Server and relay disk space; gather caches and action history can grow significantly on busy servers.
- Review DSA replication lag if running a warm-standby secondary server.
- Periodically review and prune stale custom content and unused computer groups to keep console navigation and relevance evaluation efficient.

9.2 Common Pitfalls

Symptom	Likely Cause
"Singular expression refers to nonexistent object"	A singular inspector matched zero or multiple objects; wrap in exists or switch to the plural form.
Relevance evaluates correctly in QnA but Fixlet never goes relevant	Site subscription or computer scope excludes the target; or content hasn't gathered yet.
Action stuck at "Not Evaluated"	Client isn't reporting — check relay connectivity, client service state, and certificate validity.
Version comparison behaves unexpectedly ("10.0" < "9.5")	Comparing as raw string instead of casting to version type.
Task keeps re-appearing as relevant	Success Criteria/Relevance is testing the wrong condition, or the underlying state is genuinely reverting.

9.3 Authoring Best Practices

- Guard OS family/version early in a Relevance clause before evaluating platform-specific inspectors, to avoid errors on out-of-scope clients.
- Push conditions into `whose` and scope with `of` rather than pulling large sets client-side and filtering after the fact.
- Prefer casting types explicitly (`as version`, `as lowercase`) over relying on implicit string comparison.
- Keep custom content in dedicated operator sites, not the Master Operator/Action site, to keep authoring rights and audit trails clean.
- Always test new Relevance in the QnA tool or online evaluator against representative clients before deploying broadly.

Appendix A — Quick Reference

A.1 Content Type Cheat Sheet

Type	Re-evaluates after run?	Typical use
Fixlet	Yes — relevance is re-checked, drops off list when fixed	Compliance / vulnerability state
Task	No — Success Criteria used instead	One-time or recurring operations
Baseline	Inherits component behavior	Ordered multi-step rollout
Analysis	N/A — reporting only	Inventory / dashboards / targeting data

A.2 Relevance Quick Syntax

```
exists <expr>              true/false, no error if absent
<set> whose (<cond>)       filter a set
if <c> then <a> else <b>   conditional
<x> as <type>              cast
<a> AND <b> / <a> OR <b>   boolean combination
NOT <expr>                 negation
<str> contains <sub>       substring test
plural forms: files, services, keys, processes ...
```

A.3 Useful Console Shortcuts

Task	Where
Test a Relevance expression	Tools menu → QnA, or Fixlet Debugger
View audit trail	BigFix Administration Tool → Audit Log
Manage operators	Domain Panel → All Content → Operators (Master Operator only)
Enable an external site	License Overview dashboard, or Manage Sites node
Build an automatic computer group	Right-click Computer Groups → New Automatic Group

Appendix B — Official Sources & Further Reading

This guide is a compact, independently written companion — not a substitute for HCL's official documentation, and not a reproduction of it. For exhaustive, authoritative, and version-current detail (especially the full inspector library, which runs into the thousands of entries), use these official sources:

Official BigFix Platform 11.0 guides (PDF)

help.hcl-software.com/bigfix/11.0/platform/Platform_pdfguides.html

- Getting Started Guide
- Installation Guide
- Configuration Guide
- Console Operator's Guide
- Asset Discovery User's Guide
- Web Reports Guide
- Explorer Guide
- Reports Guide
- WebUI User's Guide
- WebUI Administration Guide

Relevance Language & Inspector Reference (web-based, searchable)

developer.bigfix.com/relevance/ — guides, tutorial, full inspector search/reference library, and an online evaluator sandbox.

Support & Community

- BigFix Support Portal — support.hcl-software.com
- BigFix Forum (community Q&A)
- BigFix Playlist / Tech Advisors channel on YouTube

Compiled July 2026 as an original working reference. Not affiliated with, endorsed by, or a reproduction of HCL Technologies Ltd. or IBM publications.