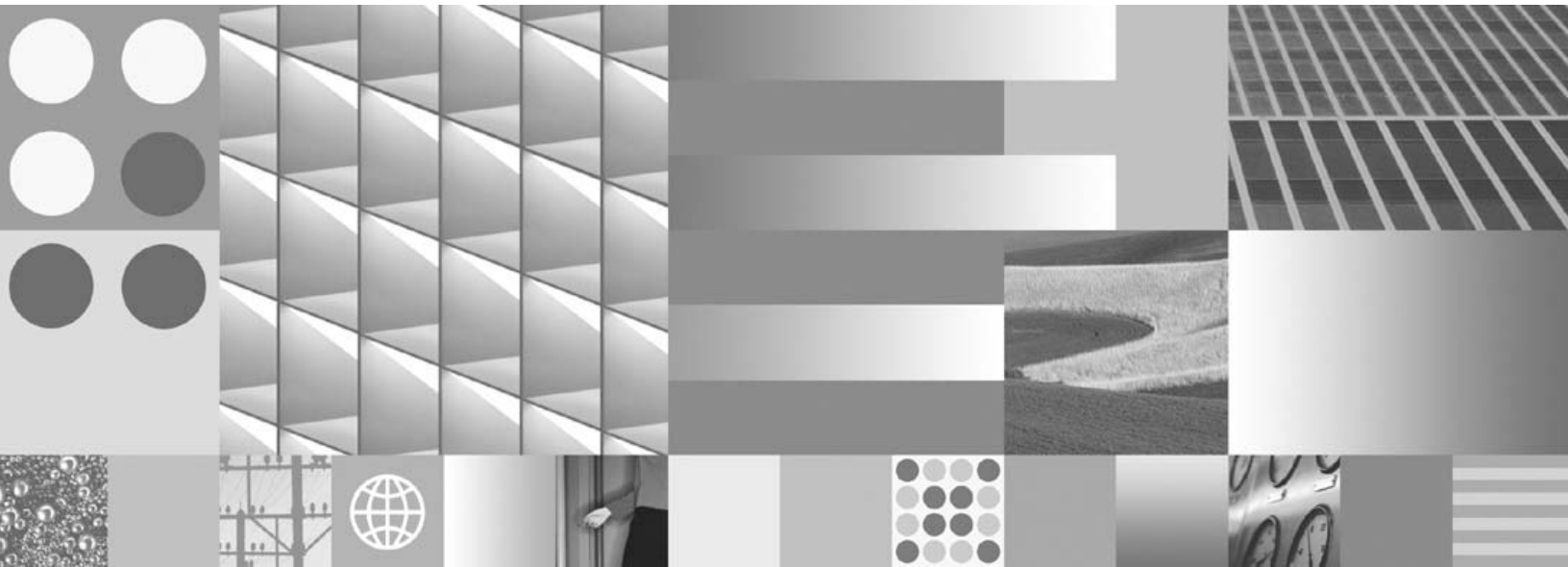


System Overview



System Overview

Note

Before using this information and the product it supports, read the information in “Notices” on page 77.

This edition applies to version 4.0.0 of IBM FileNet Business Process Framework (product number 5724-R75) and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright International Business Machines Corporation 2002, 2007. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

High-level architecture	7
Overview	7
Case management	8
Domain object model	8
Physical model	8
Content Engine object	9
Workflow object	10
Abstraction model	10
BPF Case management concepts	10
BPF tools and components	13
Added benefits	14
Data integration	14
Architecture	17
Data Tier	18
BPF Business Tier	18
Client Tier	20
Security model	21
Authentication	21
Authorization	21
Extensibility	22
Internal architecture	23
Overview	23
Case object physical model	23
Attachments	24
Audit logging	25
Workflow object	26
Case object abstraction model	26
Object model	27
Security model	29
BPF Business Services	29
Concurrency	29
Pessimistic concurrency	29
Optimistic concurrency	30
Lock Manager	31
Inbasket Manager	32
Case ID Manager	33
BPF Cache Manager	34
Inbasket Case count	35
Database access provider	35
BPF Business Services Java API	35
BPF Web Application	36
Architecture	36
Workplace integration	36
Session management	36
Action Dispatcher	37
IsDirty	37
Lookup field	38
BPF Web client	39
Sign In	39
Main Page - BPF Web Application	40
Case User Interface	41
Tools	42
Object Viewer	44

BPF Search.....	45
User preferences	46
Bulk processing.....	46
Sign Out	47
Localization	47
BPF Business and Client Tiers extensibility	47
BPF Web Application tools	47
BPF Web Application tabs	47
BPF lookup handler	48
BPF Web Client JavaScript API.....	48
BPF Business Services Java adaptor interfaces.....	48
BPF Web Application user preferences.....	49
BPF Web Client user interface customizations	50
BPF configuration	51
Case User Interface.....	53
Case fields	53
Tabs	53
BPF Search	54
Globalization and localization.....	54
Globalization	54
Localization	54
BPF Explorer	56
Architecture.....	56
Features and functionality.....	58
BPF Operations - Component Integrator Component.....	62
Architecture.....	62
Logging	62
Create object.....	62
Flow of events.....	62
Create Case.....	63
Update Case	65
Attach document.....	66
Log event	67
Deployment	68
Packaging	68
Deployment types	70
Versioning	71
Error and trace logging.....	72
Error logging via Log4j.....	72
Trace logging via AspectJ.....	74
Action dispatcher logging.....	75
Utilities	76
Workflow Region Maintenance tool	76
Workflow Map Import tool	76
Notices	77
Trademarks	78

High-level architecture

Overview

NOTE This document has not been updated for any new or changed functionality in BPF version 4.1.0. For information on new and changed functionality, see the *Business Process Framework New Feature Addendum*.

IBM® FileNet® Business Process Framework (BPF) is a case management framework. It operates with complex business objects as opposed to BPM, which operates with workflow objects, or ECM that operates with documents, when manipulated with Workplace. A BPF business object (referred to as a Case), is a collection of custom objects in a Content Engine (CE) repository, which include the Case object itself, attachment objects (documents, folders, etc), audit log event objects, etc. A BPF Case may or may not be related to one or more workflow objects. BPF is always operating with a Case, although it could be configured to mimic BPM step processor behavior. BPF presents the business object in a consistent and intuitive manner, hiding the complexity of the different components stored and managed in distinctly different repositories, allowing for synchronization of the Case, its documents, and workflow. BPF transactions are significantly more complex than the typical BPM or ECM transactions, as one BPF transaction initiates multiple CE and Process Engine (PE) object transactions.

BPF is also a IBM Lab Services delivery approach that employs consistent, reusable code and tools that IBM Lab Services use to create custom case management applications.

BPF includes:

- Consistent and configurable web user interface for both private and public Inbaskets
- Configurable case user Interface for work presentation
- Case search capabilities
- Configurable plug-in architecture for extendibility
- BPF Explorer – A comprehensive configuration tool
- Complete CE/PE integration to fully leverage all FileNet P8 Platform functionality

This document describes the BPF architecture in detail.

Case management

BPF as a case management solution provides a framework for the development and deployment of business solutions based on the integration of content and workflow management. The central concept behind a case management solution is the Case. CE provides content management services, the PE provides business process services; the Case file provides integration between the two.

BPF Case management:

- Is based on J2EE platform
- Is CE/PE centric
- Stores the Case data in the CE object store
- Leverages the object management capabilities of CE to provide more flexible interactions between CE and workflow
- Leverages the CE API to enforce standardized integration with LDAP
- Leverages the BPM Component Integrator for Case related background processes

The key differences between BPF and the Workplace tasks are:

- Added Case management control
- Entity of a Case (collection of data, documents, and events) is kept intact even after the work is complete
- Adds new documents or attaches existing documents or folders from CM repository
- Requirement to push work to users
- Role based work distribution and work management requirements
- Complex prioritization and sorting criteria
- Effective and readily available logging of system events
- More flexible user interface functionality (pick lists, validations, response driven events, etc.)

Domain object model

The BPF domain object model relies on CE and PE object models. BPF utilizes the PE repository to store the process related data and CE object repository to store the business object related data. BPF provides a metadata abstraction layer that integrates CE and PE object definitions and extends the domain object model with additional features.

Physical model

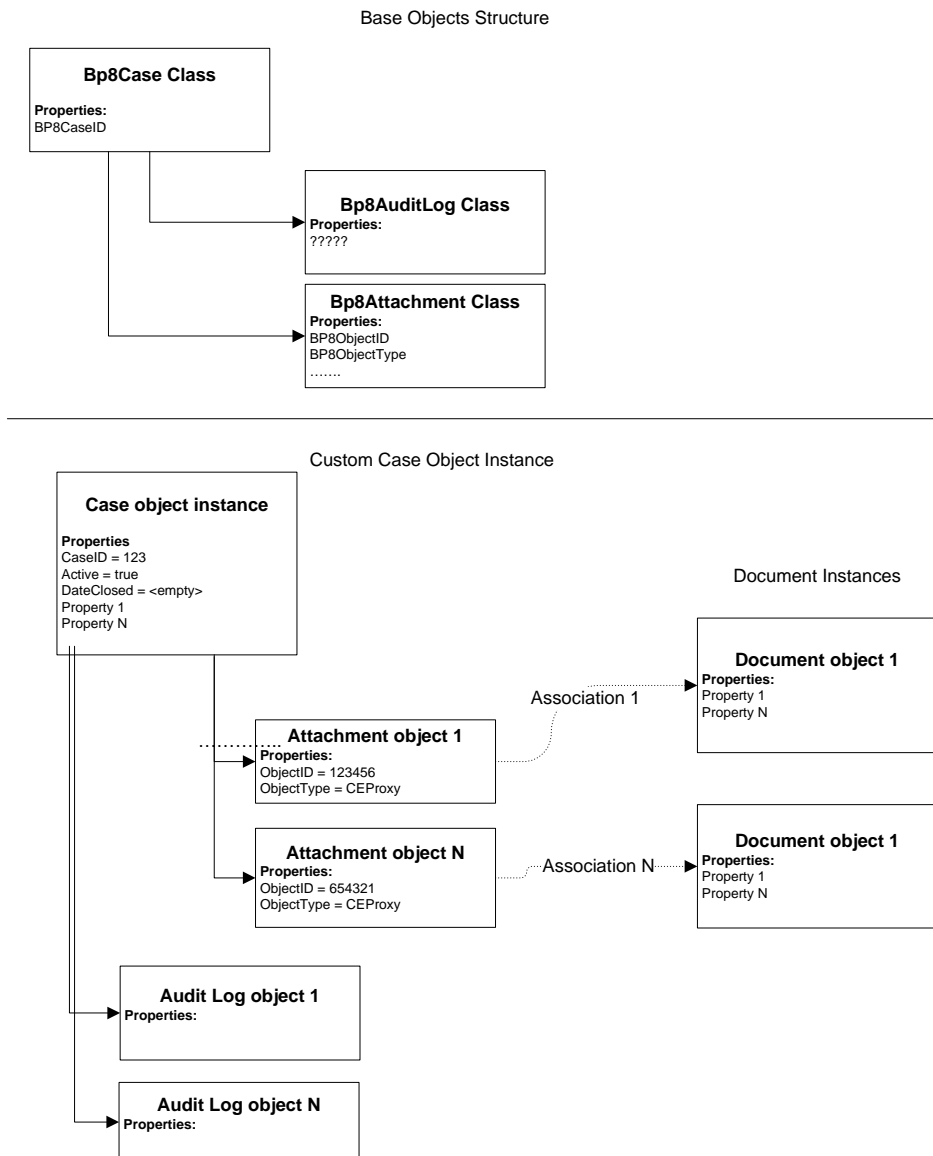
The physical model describes where and how the Case object data is stored.

Content Engine object

CE services provide content management for the BPF case management applications. A BPF Case object comprises the following CE classes and objects:

- A Custom Object for all case related information
- Document Objects for attached content

All BPF object classes are inherited from the `Bp8Object` class. The BPF Case is based on the CE's `Bp8Case` class (and derived classes). The `Bp8Case` object acts as a container for content and active workflows. An instance of the `Bp8Case` class and its associated objects implement a Case. Each Case is assigned a unique Case ID. The Case object structure is shown in the following illustration.



In this release, a single BPF application is supported with a single CE object repository only.

Workflow object

A BPF Case can relate to zero or more workflows. A Case object is associated with the workflow object(s) via the Case ID. Each defined workflow utilized in a BPF case management solution must have the Bp8CaseID field defined. BPF generates the Case ID when a Case object is created and synchronizes the field values for the CE and PE objects.

It is possible for a Case to have multiple active work objects associated with it. This could be due to a workflow object split or a parallel workflow process. Each workflow object should have the same Bp8CaseID field value to be linked to the Case object.

In this release, a single BPF application supports a single workflow region in a single server configuration, e.g. all workflow queues must reside in the same PE database.

Abstraction model

The abstraction model describes an enhanced Case object model that serves as a facet for the external clients utilizing the BPF Business Tier.

The BPF metadata provides this abstraction layer, which allows extending and customizing of the object model and its visualization based on the business requirements. All BPF metadata is stored in the BPF Metastore database and is managed via the BPF Explorer tool.

BPF Case management concepts

User

Users are defined in an LDAP enterprise directory BPF caches user account in BPF Metastore upon user logon to extend it with the following attributes:

- Default security profile
- Supervisor's name
- Department level membership
- Company level membership
- PE user ID
- User-defined attributes

Role (security profile)

A user's membership in BPF security profiles defines a user's responsibility within the context of a BPF application. User accounts are created in LDAP and are assigned to LDAP security groups. LDAP groups are used to secure Workplace access roles. Workplace access roles map to BPF security profiles. BPF security profile implements an extended set of attributes to LDAP security groups as it relates to case management and define the following:

- LDAP group it is associated with
- List of Inbaskets associated with this profile and their display order
- Various default parameters for the BPF Web Application

A user can belong to one or more security profile.

Case

A BPF Case represents an individual business transaction and is an instance of the `Bp8Case` custom object. It is comprised of the following objects:

- Zero or more PE work objects
- Zero or more `Bp8Attachment` attachment objects in a repository
- One or more `Bp8AuditLog` objects in a repository

Case field

A BPF Case field represents transactional data stored with the case and is defined by the following:

- **Field name** – Can be referenced via the BPF Web Application.
- **Display label** – As displayed in the BPF Web Application.
- **Field type** – Defines the presentation and format of the field. Field types include:
 - String
 - Integer
 - Money
 - Date/time
 - Boolean
 - Workflow user group
- **Pick list** (optional) – An associated list of values for the Case field.
- **Name of the CE Case custom object property** – The Case field value will be written to this property when the Case is updated.
- **Name of the CE document class property (for attachments)** – The Case field value will be written to this property when the Case is updated.
- **Name of the PE workflow object property** – The Case field value will be written to this property when the Case is updated.
- **Field data source** – Defines whether the Case field value is read from the CE or PE property when the Case is open.
- **Visualization parameters** – Number of rows, maximum number of characters, etc.

Pick list

A pick list is a collection of predefined property values that present a list of valid choices. A pick list can be assigned to one or multiple Case fields. A pick list typically is comprised of ID, Code (short value), Description (longer description), and an Active Flag. A pick list can also contain additional user defined columns. A pick list can be a static list, or a dynamically generated list defined via a SQL statement.

Pick list values can be localized (native language support).

Case type

A Case type is a collection of attributes and features that uniquely identify a Case object type that implements a business object for a specific business function. A Case type is comprised of the following:

- A unique set of Case fields
- Default CE Case object class name
- Default CE attachment document class name
- Default workflow name

Default values for the fields indicated above could be overwritten using the BFP Business Services interfaces.

A Case type defines the unique user interface layout for the Case object. BPF allows you to modify the Case object's Case type during runtime, which allows for re-classification of the object without changing its object class in CE.

Inbasket

An Inbasket is a feature that allows locating and processing BPF Cases in the context of workflow. Inbaskets in the BPF Web Application provide an interface that delivers a browseable, filtered access to workflow queue(s) based on the user role. Inbaskets provide access to Case objects rather than workflow objects. A workflow queue, step, and security profile are always needed to configure an Inbasket. This means that there are as many Inbasket definitions as combinations of work queues/steps and roles.

Inbaskets are defined by the following:

- Filtered view of workflow queue contents
 - **Queue Filters** – These configured filters determine what the user can see in the queue.
 - **Inbasket Filters** – These are configurable filter options that the user can invoke to further limit the Inbasket list.
- Sort order
 - Sort order prioritizes work delivery for users and can be based on a single field, a combination of fields, or a complex SQL custom function.
 - User column sorting is configurable, each column supporting complex sorting criteria.
- Case browse list columns
- Case level functions
 - Tools that can be exposed to the user depending on the role (Add document, Create New Case, etc.)
- Case level actions
 - Actions map to the responses found on the process map.
 - Actions may have required specific fields be filled in.
 - Actions may prompt for additional input or reasons.
 - Actions can have custom code plugged in for execution.

- Case fields and their presentation
 - Case layout according to Case type
 - Expandable/collapsible zones of fields
 - Required fields
 - Read only fields
 - Reason lists
 - User pick lists
 - Field assignments
- Case access mode (Browse vs. Get Next)

The BPF Inbasket allows you to extend the Case abstraction data model via custom plug-ins to merge and present the data received from external host systems. Data could be gathered from multiple sources for display in a single user interface.

BPF tools and components

BPF Business Services

BPF Business Services is a set of modules and services that implement all case management business logic.

BPF Web Application

The BPF Web Application is a user interface that provides access to work by users. The BPF Web Application's user interface is highly configurable and is generated dynamically based on a user's role.

BPF search

BPF search is a function within the BPF Web Application that allows locating and opening of BPF Cases outside the context of a workflow. BPF search is based on the CE stored search templates. The user's access to BPF search templates and the Case object instances is controlled via the CE Access Control List (ACL).

BPF operations

The BPF operations are Component Integrator-based Java™ adapters that are designed to provide a number of case management functions, including: Create Case, Log Event, Update Case, etc.

BPF Explorer

BPF Explorer is a configuration tool used to define and manage the BPF abstraction layer.

BPF Metastore

The BPF Metastore is a Microsoft® SQL, Oracle, or IBM DB2® database used to store all the configuration definitions described via BPF Explorer. This data is cached in the Metastore cache during runtime and is used by the BPF Web Application to control user's access to business objects and process work.

BPF manifest

The BPF manifest is an XML based export of BPF configuration. The BPF manifest allows for backup and restore of the BPF configuration. This is typically necessary to move the configuration between systems as well as for version control, to backup a version because new changes are made.

Added benefits

Case ID generation

Each Case object within BPF has a unique Case ID generated by BPF and is an auto-incremented number.

Localization

The BPF visualization layer (Web Application) is fully localizable in UTF-8 languages.

CE and PE data synchronization

BPF synchronizes the Case object property values in CE and PE repositories when the Case is updated and emulates the transaction commit and rollback.

Data integration

Object relationships

- Each BPF Case always has an instance of the CE `Bp8Case` custom object class.
- A BPF Case may have zero or more workflow objects. Work objects could be split in workflow and processed in parallel.
- A BPF Case may have zero or more attachments (documents or folders).
- A BPF Case may have zero or more audit log objects.

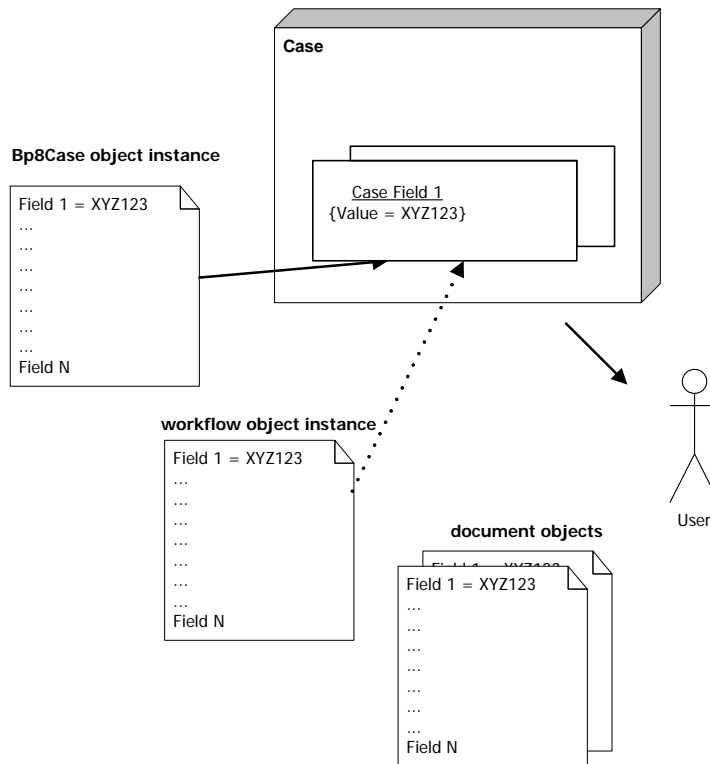
Opening a Case

When cases are browsed in the Inbaskets, workflow queues supply the Case ID and the case information is read from the Case object for display.

When a case is opened, either from the Inbasket or from a BPF search, the Case field data could be read from one of the following data sources, according to each individual Case field configuration:

- `Bp8Case` custom object property
- Work object property
 - If there is no workflow object associated with the Case, field values will be read from the `Bp8Case` object property

When a Case is opened from a BPF search, the values displayed in the Case fields are always from the work object properties. If there are multiple workflow objects, the data will be read from the first available work object. The objects and their relationships are shown in the following illustration.



Completing a Case

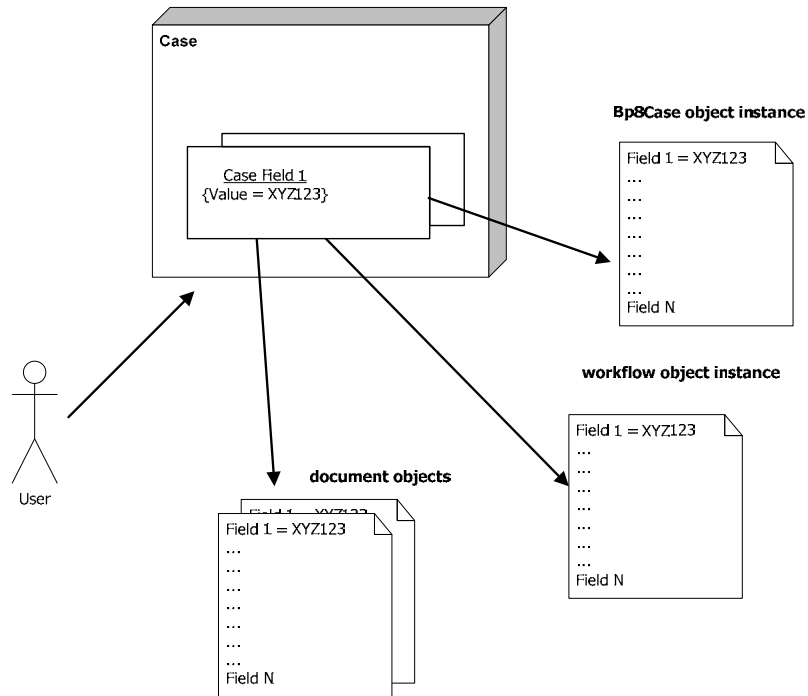
When a Case is saved or completed from the Case user interface, Case field data could be saved to any or all of the following data sources, according to each individual Case field configuration:

- Bp8Case custom object property
- Workflow field property
- Attached document(s) property
- Related audit log objects (this is not part of the case metadata, but is a related set of objects representing events)

There is always only a single Bp8Case object related to the Case. BPF will manipulate the Bp8Case object that the user is working with and attempt to update all the property values that have been configured as CE properties in BPF Explorer. If errors are encountered when trying to update these object properties, they will be logged in the BPF error log file and the operation will be aborted.

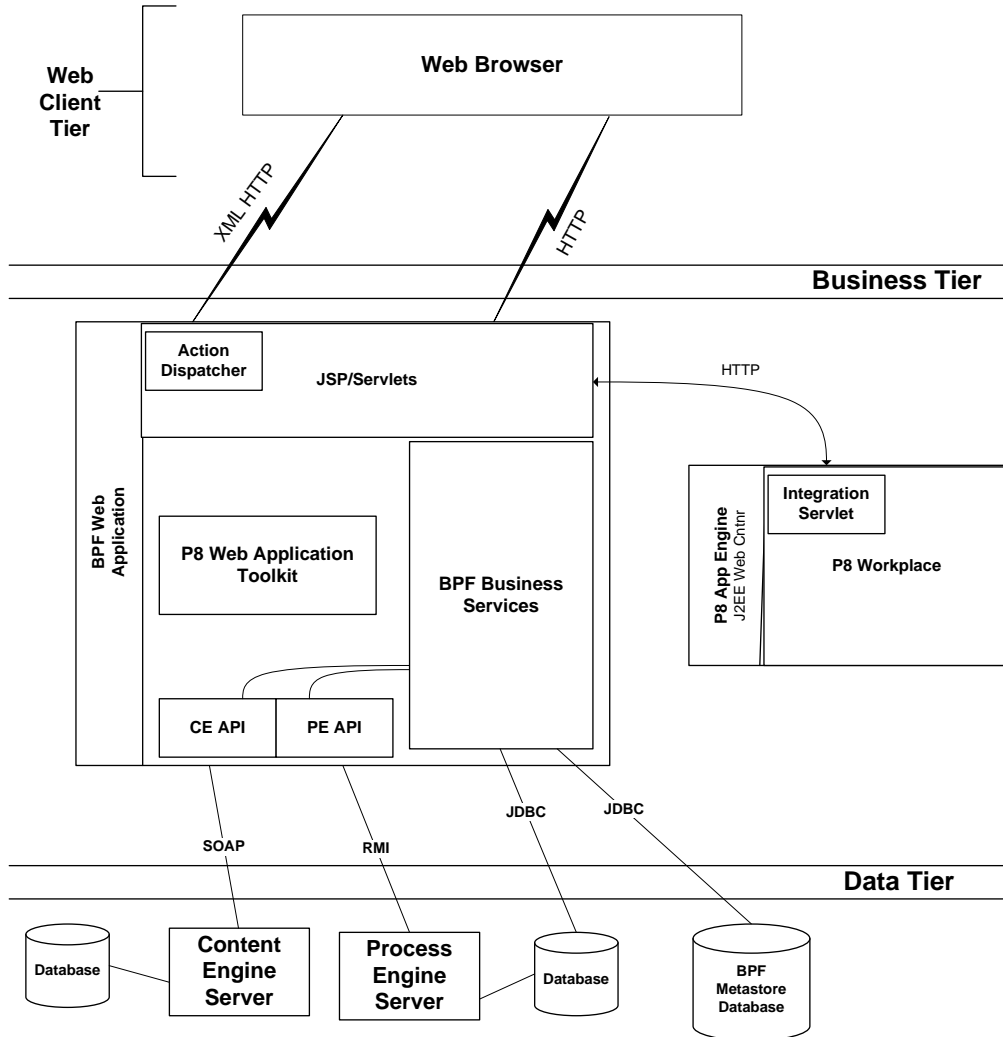
Because there may be multiple attachments on the Case, BPF will attempt to update the property values for each document that has been configured as attachment field properties in BPF Explorer. A Case might have documents with various document classes attached. If any access related errors or property related errors occur when updating these document properties, they will be logged in the BPF error log file and operation will continue.

There could be multiple workflow objects related to the Case. BPF will manipulate the work object that a user is working with and attempt to update work object property values that have been configured as PE properties in BPF Explorer. If errors are encountered when trying to update these workflow object properties, they will be logged in the BPF error log file and operation will continue. This is to support different versions of the workflow process map. If any errors are encountered during workflow object dispatch, the entire transaction will be aborted and BPF will attempt to roll the `Bp8Case` object data back. The objects and their relationships are shown in the following illustration.



Architecture

The BPF Architecture is shown in the illustration below.



The following sections will discuss each of the tiers of this architecture in detail, moving from the data tier up through the client tier.

Data Tier

The Data Tier refers to the persistence layer, core FileNet P8 Platform services, and the third-party services that serve as a repository for the BPF domain object model and configuration.

CE

BPF utilizes a CE repository for storing and managing the business object data. The BPF Business Tier uses the CE Java API for all content related operations.

PE

BPF utilizes a PE repository for storing and managing the business object process state. The BPF Business Tier uses JDBC to retrieve workflow queue content and the PE Java API for all process related operations.

BPF Metastore

BPF Web Application settings, domain object model, and data visualization definition are stored in a dedicated BPF Metastore database. The BPF Metastore database may be collocated with the PE or CE database server or be installed on a separate database server. The BPF Business Tier uses JDBC to communicate with the BPF Metastore.

Scalability

The Data Tier can be scaled up using multiple CPU servers. The following constraints will apply:

- The BPF Metastore is scaleable across multiple servers. However, this option is unlikely since most of the relevant BPF metadata is cached by the BPF Business Tier at runtime, eliminating traffic to the BPF Metastore.
- The PE scalability is subject to the PE version-specific features and limitations.
- CE scalability is subject to CE version-specific features and limitations.

BPF Business Tier

For the purpose of this section and those that follow, the BPF Business Tier is sub-divided into two parts: the BPF Web Application and BPF Business Services.

The BPF Business Tier provides a structural model for the application data and a set of procedures for performing business related activities. The BPF Business Tier:

- Is shared by multiple clients
- Provides a framework for case management transactions, ensuring that the information in the case model remains in a valid state
- Provides a persistence mechanism for application data

BPF Business Services

BPF Business Services are a set of modules that sit above the Data Tier and encapsulate the case management business logic. In this release, BPF Business Services are not hosted as an independent service. The reason for separating the BPF Business Services in this document is to demonstrate that this component could be utilized directly by various clients as well as point out future architecture goals.

BPF Business Services expose a set of high-level functions that allow the clients to manipulate the BPF business objects:

- Create, retrieve, update, and delete

- Manipulate the process state
- Perform event logging

BPF Web Application

The BPF Web Application provides the Inbasket and work presentation user interface and is one of the core BPF elements. The BPF Web Application is hosted within a J2EE application server environment and utilizes BPF Business Services. The web browser-based client communicates with the BPF Web Application directly or through a load balancer.

The BPF Web Application contains a feature-rich browser-based user interface, with nearly thick client functionality. The architecture is optimized for scalability and contains a myriad of user productivity features. In order to implement this feature-rich user interface functional requirements and to leverage as much FileNet product functionality as possible, BPF attempts whenever possible to reuse and extend upon the user interface functionality provided by the FileNet P8 Platform Web Application Toolkit and Workplace as opposed to designing and creating a custom Web Application using a from scratch approach.

P8 Workplace is a J2EE application built on top of the Web Application Toolkit (WAT). WAT is a reusable components framework for building custom applications. It provides API level interfaces that are supported by IBM engineering.

The overall design direction was to utilize the CE and PE API directly where applicable, the WAT API where possible, reuse the Workplace Integration Servlet-provided user interface elements where available, and develop any additional custom code necessary to meet the functional requirements.

The BPF Web Application is built around an AJAX (Asynchronous JavaScript™ and XML) development model, also utilizing the FileNet P8 Platform Web Application Toolkit (WAT). The BPF Web Application exposes Case Management services via an Action Dispatcher servlet. The Web client uses XMLHttpRequest (AJAX) for communication with the Action Dispatcher. The design of the BPF Web Application minimizes the client-server traffic by updating the browser only with the data that has changed. The BPF Web Application also contains a set of client-side JavaScript functions that employ data validation techniques, minimizing unnecessary round trips to the Web server.

The BPF Web Application user interface is metadata driven and is built dynamically upon user request. The metadata describing the user interface layout is stored in the BPF Metastore database and is managed via the BPF Explorer configuration tool.

When a user makes a request to retrieve or manipulate a business object:

- The business object presentation manifest and any associated business rules are retrieved from cache or the BPF Metastore (if the data is not present in cache).
- The business object data is retrieved from the PE and CE.
- Both data and presentation manifest are packaged as an XML message.
- The XML message is sent to the browser.
- The browser will render the HTML user interface via the client side XSL transformation and make the business object available to the user.

Action dispatcher servlet

The BPF Web Application services are exposed as an action dispatcher, which is a passive listener accepting and processing requests from the web browser clients. The action dispatcher accepts the services requests and provides a response back as XML messages over HTTP based protocol.

Supported application servers

The BPF Web Application was designed using standard J2EE technologies and should work in any J2EE compliant application server environment. See the *IBM FileNet Business Process Framework Hardware and Software Requirements* document for supported application servers.

Scalability

The BPF Business Tier can be scaled up using multiple CPU servers or multiple servers using J2EE clustering. When J2EE clustering is utilized, clients of this tier see a single server, with a virtual IP address. This IP address is mapped to multiple physical servers via a load balancing (either hardware or software) solution. The overall system can scale to handle an increased load, by adding additional servers to the cluster.

The BPF Business Tier is not entirely stateless, requiring a sticky connection to be maintained to a specific server. The load balancing solution should support the sticky connection feature.

High availability

High availability in the BPF Web Application Tier is achieved by creating a cluster with multiple servers. If a single server goes down, the load balancer will detect this condition and stop routing requests to this server. Access to the cluster is routed through the remaining servers automatically.

Although the BPF Web Application is not entirely stateless, it provides a means to address high availability requirements by storing a snapshot of the session state on the client. When the Web client determines that the Web session is lost, which is typically due to Web session expiration, the Web client recreates the session and allows the user to continue working after logging back on to the BPF Web Application.

Client Tier

The two clients of the BPF Business Tier provided with this release are: the browser-based Web Client and the BPF Operations.

Web client

The BPF Web client provides a nearly thick client feature rich interface to the BPF Web Application and is comprised of the HTML, JSP, JavaScript pages, and other objects. The BPF Web client utilizes HTTP and XMLHTTP to access the BPF Web Application services. The Web client contains a set of client-side JavaScript functions that employ data validation techniques, minimizing unnecessary round trips to the web server.

Supported Web browsers

The BPF Web client was designed to work with Microsoft Internet Explorer.

BPF operations

BPF operations are a set of functions that are packaged as a Component Integrator-based Java class. BPF Operations create and manipulate case objects by utilizing BPF Business Services from a step on the workflow map.

Java applications

Customers may build their own Java applications by leveraging the BPF Business Services methods.

Security model

The security model employed by BPF utilizes users and groups found in the customer's LDAP enterprise directory as well as Workplace access roles.

Authentication

Each potential user of BPF is expected to have an account defined in the enterprise LDAP. BPF uses the CE and PE Java API to validate the user credentials through the authentication provider. BPF does not contain any user authentication services.

Authorization

When an authenticated user interacts with the BPF Web Application server, various authorization mechanisms are used to determine what level of access, if any, that user has to create, view, or manipulate objects within the system.

System access

Users are assigned to an application level security group in LDAP that controls access to the BPF Web Application. If the authenticated user is not part of the application level security group, access to the BPF Web Application will be denied.

Role-based authorization

Users are assigned to a role level security group in LDAP that controls which of the BPF Web Application features are available to the authenticated user. Each LDAP role level group is associated with the security profile in BPF, which defines the set of Inbaskets that define the type of work the user can process and the tools and actions available.

Access control list authorization

Once a user is granted access to the BPF Web Application, permission to the individual business objects is controlled via the user or group-based Access Control List (ACL). The business object ACLs are configured via the Enterprise Manager.

Extensibility

The BPF Business Tier exposes various plug-in Java interfaces allowing the customers to change the default framework behavior and to seamlessly integrate their unique business logic. These interfaces are described in further sections.

Internal architecture

Overview

The internal BPF Web Application architecture is based on the assumption that it will operate inside of a J2EE application server. An application server provides containers where the Web components run. These containers manage services for components such as transactions, multithreading, and resource pooling. These services are not the responsibility of components running inside of a container.

J2EE application server vendors provide tools to perform administrative tasks for containers and their related components. The services provided by the containers are part of this administration. For example, the configuration (deployment descriptor) for a container component may define transactional behavior.

Some additional benefits provided by the application server are as follows:

- JDBC connection pooling
- Multithreading

Case object physical model

The CE object store is used to store the BPF Case object. Bp8 Objects follow this class structure:

Custom Objects (standard FileNet P8 Platform container class)

Bp8Object base object

Bp8CaseID

.....

Bp8Case object

Bp8AuditLog object

Bp8Attachment object

The `Bp8Case` base object is used as a template only and not intended for direct invocation. Each implementation will create its own set of classes (loan, invoice, etc) inherited from the `Bp8Case` base class. Each custom class will implement its own set of properties. A single Case object class in CE could have multiple BPF Case types associated with it.

The `Bp8Case` base object is a child of the `Bp8Object` class and has the following standard properties, in addition to system properties inherited from the base Custom Object class:

Class	Custom Properties		Type
	Name	Description	
<code>Bp8Object</code>		Base Bp8 class that all business objects have to inherit from.	
<code>Bp8Case</code>	<code>Bp8Active</code>	Indicates whether Case object is active in the workflow. When the object is created, it is set to True. It is the responsibility of each implementation to set it to false upon closing the Case. The suggested method for setting the Case as closed is a Component Integrator step.	Boolean
	<code>Bp8DateClosed</code>	Reserved for future use.	DateTime
	<code>Bp8Comment</code>	Reserved for future use.	String
	<code>Bp8CaseType</code>	This field identifies the Case type for the Case object.	Integer
	<code>Bp8ObjectID</code>	This is the custom object GUID that identifies the case.	ID (GUID)
	<code>Bp8CaseID</code>	Auto-incrementing Case ID used for easier identification. PE work items will be linked to Bp8 Cases using this property. <code>Bp8Settings</code> object instance is used to keep the last auto incremented value.	Integer

Life cycle

A BPF Case object is typically created via BPF Business Services; however, it could also be created directly via the CE API. The BPF Case object remains in the CE object repository until deleted. All linked Case object references will be deleted when a parent object is deleted. That is, all `Bp8AuditLog` and `Bp8Attachment` objects will be deleted when the `Bp8Case` object is deleted.

Attachments

The BPF object model allows associating other documents and objects with the Case and presenting those objects to the user as attachments.

Attachment types

The BPF object model allows attaching CE folder and document objects to the Case. The visual representation of folders and documents from within the case is implemented via the Workplace Integration Servlet.

The folder content is displayed in a web page utilizing the Workplace Integration Servlet, which allows browsing a subfolder, viewing content, and performing any content-related operations.

The document objects are displayed in a viewer registered in Workplace for the appropriate mime or object type. BPF utilizes the Workplace Integration Servlet to invoke the Viewer.

Physical links

The default implementation model for attachments links each attachment to a Case via a special custom object in the CE repository, which associates the content object with the Case. One document could be attached to multiple Cases.

This model allows for a transparent update of the attachment property fields when the Case is updated.

The attachment link object is implemented via the `Bp8Attachment` custom object, which is a child of the `Bp8Object`. The following table provides the `Bp8Attachment` object definition:

Property	Description	Data type
<code>Bp8CaseID</code>		Integer
<code>Bp8ObjectID</code>	Object ID in its repository. For CE documents, this is an object id (GUID).	String
<code>Bp8ObjectType</code>	Pointer to an internal structure: 1. CE document 2. CE proxy document 3. CE Folder	Integer
<code>Bp8ObjectClass</code>		String
<code>Bp8ObjectRepositoryID</code>		String

Dynamic attachments

In this model, there are no attachment link objects that associate the document with the Case(s). The Web Application determines the Case attachments, which are based on a SQL search predefined in BPF Explorer at runtime when the user opens the Case. It is possible to have both physical link object attachments and a dynamic attachment defined for the case if necessary. This is configurable via the BPF Explorer configuration tool.

Audit logging

The Case audit log stores the history of all actions performed on the Case object. The audit log is updated automatically when a Case is opened or updates. The level of logging and action names are configurable via the BPF Explorer tool.

The following table provides the `Bp8AuditLog` object definition:

Property	Description	Data type
<code>Bp8ActionTime</code>		DateTime
<code>Bp8Action</code>		String
<code>Bp8CaseID</code>	Association property will be used here to enable referential integrity. The FileNet P8 Platform will delete all Audit Tab Item records if the main Case is deleted.	ID
<code>Bp8UserName</code>	Logon name of the user.	String
<code>Bp8UserSID</code>	SID identification of a user generated the event.	String
<code>Bp8EventCategory</code>		Integer
<code>Bp8Description</code>		String
<code>Bp8Reason</code>		String

Workflow object

BPF utilizes BPM for all process-related operations on the Case. Each BPF Case may have one or multiple workflow objects representing the Case state on the workflow map. The Case object is linked to the workflow object via the `Bp8CaseID` field. The workflow object has a link to the Case object in the `Bp8Case` attachment field.

The following constraints must be considered when designing a Bp8 workflow map:

- **Bp8CaseID (Integer)** – Must be present on the map and exposed to every step in the workflow. This field is user to link the work object back to the `Bp8Case` object. This field should also be exposed in each workflow queue and workflow roster.
- **Bp8Case (Attachment)** – When defined on the workflow map, it will contain references to the `Bp8Case` object. The FileNet P8 Platform Component Integrator will be used to manipulate properties of the `Bp8Case` object to change `Bp8Case` fields, modify Case status, mark a Case as closed, etc.
- **Bp8Attachment (Attachment array)** – When defined on the workflow map, it will contain references to the Case object attachments.
- BPF uses workflow responses to dispatch workflow objects. Reassign and Return to Queue functions are not supported in this release of BPF.

Case object abstraction model

The BPF Case object model definition is stored in the BPF Metastore.

The abstraction model defines the following:

- Case business object structure
- Integration between CE and PE
- Visualization (display layout) of the Case
- Runtime behavior based on current Inbasket and role

Object model

BPF Metastore

BPF Metadata constitutes the data that keeps track of security profiles, Inbaskets, Inbasket layout, system-wide configuration parameters, etc. A dedicated BPF Metastore database is used to store such data.

In addition to the user interface configuration data, the metadata database will contain user account extended information. BPF requires storage of some extended user information, such as: user's manager, security profiles, workflow user ID, companies, departments, customer-defined attributes for integrating with host systems, etc. Ideally, this data should be kept in LDAP, but it is common for customers to prefer not to modify their LDAP schema. There is no facility in the FileNet P8 Platform to store this information either. Short term, user account records are maintained in the USERS table in the Bp8 Metadata database. This also adds an additional level of security to the solution being implemented, preventing unauthorized users from accessing the Web Application.

Case type

Case type defines a unique data dictionary and controls the following:

- Inbasket behaviors
 - Exposed Case fields and their layout in each Inbasket could be different based on the Case type.
 - Required fields for each Inbasket response could be different for each Case type.
 - Data source for each Inbasket Browse list column could be different for each Case type.
 - Configurable favorite field value, which is displayed in the BPF Web Application status bar.
 - Case layout for Case objects opened from BPF Search.
- Associated data object types
 - Default workflow name
 - Default attachment document class name
 - Default object class name

Case fields

The following metadata is stored with the BPF Case fields:

- The storage location which, based on the field definition, could be saved to or retrieved from:
 - CE field (bi-directional)
 - Workflow field (bi-directional)
 - Attachment field (save-only). At the Inbasket level, it is possible to define whether the attachment index fields need to be updated (default is off).

- Field type
 - Money
 - Numeric
 - Date/time
 - Boolean
 - String
 - Workflow group
- Display parameters
 - Name
 - Label
 - Number of rows
 - Column width
 - Maximum number of characters
- Pick list
 - Display code, description, or both
- Lookup
 - Whether the pick list should be displayed as a lookup
 - External source URL

Pick lists

A BPF pick list is a choice list of possible values that allows limiting the Case field value. The BPF Business Tier enforces the pick list when saving and retrieving object data. A pick list does not prevent you from using CE choice lists to enforce data integrity on the Case custom object level.

A pick list is typically comprised of the following columns: ID, Code, Description, and Active flag. However, a pick list could be extended with any number of custom columns. Pick lists can be defined as static or dynamic.

Static

Static pick list values are defined via the BPF Explorer configuration tool and are stored in the BPF Metastore. The ID column value is automatically generated. The Active flag allows disabling a specific value for selection. However, existing Case objects that contain an inactive value would still be able to display it.

Dynamic

A Dynamic pick list is defined as a SQL statement and the values list is rendered at runtime. The BPF Metastore is typically the data source for the dynamic pick lists.

Lookup attribute

A Lookup is an attribute to the pick list that tells the BPF Business Tier that only the current value of the Case field should be retrieved and not the entire list. This attribute is typically used to improve the retrieval performance of large pick lists.

User lists

User lists are dynamic pick lists defined against the USERS table in the BPF Metastore and contain user names and their attributes. User pick lists are typically used for workflow step assignments. User pick lists allow limiting the choice list based on certain user properties: Department, Business Unit, Approval Limit, etc.

Security model

Security profiles

Security profiles implement the role-based work distribution and define what work and functionality users have access to. Security profiles also keep track of the case-management specific attributes that could not be stored with LDAP groups or Workplace access roles:

- Inbasket list, their display order, and the set of associated properties:
 - Default Case tab
 - Whether the Inbasket Case count should be displayed
 - LDAP security group name
- All of the display characteristics in the Case user interface:
 - Case tab and case fields
 - Actions, tools, and tabs

Users can be assigned to one or multiple security profiles.

BPF Business Services

BPF Business Services is a set of modules and services that implements all case management business logic. This section describes some of the key functionality in details.

Concurrency

Pessimistic concurrency

Goal

To allow multiple users access to shared resources in an environment where the possibility of collision is high, without forcing user to discard work.

Interactions

A client connects to the BPF Business Tier and initiates a Case edit session. The BPF Business Tier invokes the Lock Manager to request a lock on the specific workflow object the user wants to edit. If the lock is unavailable, the Business Tier returns a message to the user. If a lock is available, the business delegate retrieves the data from the workflow and the Case custom objects and submits them back to the user. The user makes the edits to the Case object and submits the changes back to the business delegate that saves the data back to the workflow and the Case custom objects and releases the lock.

NOTES

- The BPF Business Tier only implements pessimistic concurrency support for Cases opened via an Inbasket. BPF Business Services unlocks and releases the workflow object when the user is closing a Case or executing an Action (workflow response). If locks are not released properly, the whole system can gradually become unavailable.
- BPF Business Services does not utilize CE object locking. When a Case is opened via an Inbasket, the CE Case custom object is not locked, enabling optimistic concurrency.

Optimistic concurrency

Goal

To allow multiple users safe access to shared resources in scenarios in which there is minimal risk that two users will simultaneously edit the same data.

Interactions

A user interacting with the BPF Business Services requests data to view or edit. If the Case is opened from an Inbasket, a pessimistic concurrency scenario is executed. If the Case is opened from the BPF search:

- If there is no corresponding workflow object, all data is retrieved from the Case custom object. The Case custom object is not locked.
 - If the Case edit is enabled, updated data will be saved to the Case custom object.
- If there is a corresponding workflow object, the data is retrieved from both the workflow object and the Case custom object. Case custom object and the workflow objects are not locked.

NOTES

The BPF Business Tier implements optimistic concurrency support via the following means:

- Cases opened via BPF search are typically defined with all fields as read-only. Multiple users can view the same Case without modifying it.
- BPF search allows exposing different sets of Case fields as read-write for members of different security profiles, allowing simultaneous read-write access to the Case custom object.
- BPF supports BPM workflow object splits and multiple step participant assignment to allow the Case to be work on simultaneously by multiple users. The Case fields that could be simultaneously edited by the users need to be defined with PE as data source with the data merge rules configured in the workflow process map.

Lock Manager

Goal

Provide a central point for managing lock information.

Interactions

When a business delegate needs to modify a Case object via an Inbasket interface, it calls the Lock Case method. If a lock cannot be obtained, the Lock Manager passes a message back to the client, along with the name of the user who holds the lock. Locks are released via the Unlock Case method or when a user executes an action on the Case.

Open or lock a Case

- When a client makes a request to open or lock a Case, it passes the Inbasket ID, Case ID, and the workflow object number to the Lock Manager.
- The Lock Manager locates and fetches the workflow object from the queue.
- The Lock Manager locks the workflow object to the requesting user and passes control back to a business delegate.

Unlock or dispatch a case

- When a client makes a request to dispatch or unlock a Case, it passes the Inbasket ID, Case ID, and the workflow object number to the Lock Manager.
- The Lock Manager locates and fetches the workflow object from the queue.
- The Lock Manager locks the workflow object to the requesting user and passes control back to a business delegate.

Implementation

The Lock Manager implements a stateless service. When the workflow object is locked, the PE VWObject is discarded and a reference to the locked object is released. A workflow object is released by a user only when executing an unlock method or dispatch. When a Case unlocks or a dispatch is requested, the workflow object is located in the queue and locked with an Override option.

NOTE Users can always override their lock but cannot override locks from other users.

Inbasket Manager

Goal

To provide configurability and flexibility to workflow queue content filtering and sorting.

Interactions

A client connects to the BPF Business Tier and initiates a request for the Inbasket contents. The request is passed to the business broker, which will determine and build the SQL statement based on the Inbasket definition. The SQL statement will be executed against the PE database workflow queue (view) and the list of work objects and their properties exposed in the queue will be retrieved. The Bp8CaseID field is read from the first *n* records retrieved from the queue and a search is executed against the CE object repository to retrieve the Case objects where the Bp8CaseID property value is equal. A list of those Cases that have a list of corresponding workflow and Case custom objects is returned to the client.

The workflow queue record set is saved in the context of the client's session to allow page navigation within the Inbasket. A client can request a first, next, previous, or a last page from the result set.

Implementation

Queue filters

Queue filters are pre-defined SQL statements that are evaluated during runtime and executed against workflow queue to retrieve content. Queue filters contain macros, which are evaluated and substituted with runtime values.

Example:

```
select %PAGESIZE% F_WobNum, F_UniqueId, F_Locked, F_LockUser, F_BoundUser, Bp8CaseID,  
%FIELDLIST% from %VIEWNAME% WHERE %FILTERBY% F_BoundUser = %USERID% order by %ORDERBY%,  
F_EnqueueTime, F_UniqueId asc
```

Inbasket filters

Inbasket filters are user interface elements associated with an Inbasket that allow a user to enter a workflow field value to locate an individual object or a set of objects.

Inbasket columns

Inbasket columns are the workflow and the Case custom object fields that are configured to be displayed on the Inbasket browse list.

Macros available to queue filters and Inbasket filters

- **%PAGESIZE%** – This macro contains the Inbasket page size used to retrieve a subset of the workflow queues content.
- **%VIEWNAME%** – This macro contains the workflow queue name (view) configured for the current Inbasket.
- **%FILTERBY%** – This macro contains the concatenated values from all Inbasket filters and is populated by a user at runtime.
- **%USERID%** – This macro contains current user's workflow user id and allows filtering work assigned to the current user only.
- **%ORDERBY%** – This macro contains the list of workflow object fields configured for the Inbasket default sort column or the column selected by the user.

Case ID Manager

Goal

To provide a user-friendly Case custom object unique identifier.

The Case ID is a unique numeric identifier that is assigned to each Case in BPF when a new Case is created.

Interactions

A client connects to the BPF Business Tier and initiates a new Case creation. BPF Business Services sends a request to the Case ID Manager to obtain the next available Case ID. The Case ID Manager retains a range of the available Case IDs. If an ID is available, the Case ID Manager removes the ID from the pool and returns it to the client. If an ID is unavailable, the Case ID Manager retrieves the range of IDs by performing the following sequence:

- Locates an instance of the `Bp8Settings` object and locks it.
- Reads the `Bp8LastCaseID` property value, increments it to the size of the Case ID range, stores the new value back to the `Bp8LastCaseID` property, and releases the lock.

Once the ID range is obtained, it is cached in the Case ID Manager and the next available ID is returned to the client.

Implementation

The Case ID Manager is an instance of the `CaseldManager` Java class that is implemented following the J2EE Singleton design pattern. The Case ID Manager enforces the integrity of the data model since the business logic is encapsulated in the executor and not in the consumer. The Case ID Manager captures all errors to a `log4j` log file.

The last Case ID used is stored in a custom object in CE and is implemented as an instance of the `Bp8Settings` class. Although the `Bp8Settings` object can be made secure by assigning object security, this makes sense only if the object is accessed via a special user account, and not within a user context. The BPF Case Manager employs the user context to modify the `Bp8LastCaseID` property value of the `Bp8Settings` object.

NOTE If there are multiple BPF applications installed on the same server, each has an instance of its own executor class. Once a range of the Case IDs is reserved, it will never be released back if unused. The unused portion of the IDs from the range could be lost in the event of a server failure, application shut down, etc. Unused Case ID numbers are never reclaimed. The Case ID reservation size is configurable as an application setting in the BPF Explorer. Typically, it is set to 1 in a development or demo type environment, and is set to 100 or higher in a production environment.

BPF Cache Manager

Goal

To avoid delays created by unnecessary and excessive database access.

All metadata describing the Case object abstraction layer is stored in the BPF Metastore database. This metadata includes Case fields and types, Inbasket definition and layout, security profiles, etc. Typically, the metadata is volatile during design and development and fairly static in production. The BPF Case Manager fetches the necessary abstraction layer components each time a Case or an Inbasket is accessed to reconstruct the necessary layer. Fetching this data directly from the BPF Metastore database each time a BPF object is accessed is quite costly and requires extra processing power from the database server hosting the BPF Metastore.

The BPF Case Manager utilizes a Metadata Cache service, which allows for caching the metadata. BPF Metadata Cache employs a lazy initialization, on-demand caching schema where each specific metadata element is cached the first time it is accessed by a user. Certain elements are however initialized during the BPF Web Application initialization.

Interactions

The client calls methods on a business object, requesting specific metadata. The user request is passed to the BPF Cache Manager. If the object is already in cache, it is returned to the user. If the object is not in cache, it is retrieved from the Data Tier, stored in cache, and returned to the user. If the client needs additional data, the business object performs an additional request to the BPF Cache Manager.

Implementation

The MetadataStore Java class is a factory for the metadata objects. Typically, the metadata objects are represented as DOM documents or serialized XML strings. The MetadataStore class has specific methods for each type of the metadata objects; the purpose of these methods is to access the database and create a corresponding cache object.

The Metadata Cache meets the following requirements:

1. All repetitive requests to the metadata must be cached.
2. Objects in the cache must be identifiable by a unique key.
3. Cache must be able to store objects of different types.
4. Metadata Cache supports localized data.
 - Bp8CacheltemImpl and Bp8CacheImpl classes implement the Metadata Cache.
 - Bp8CacheltemImpl is a wrapper for all objects stored in cache.
 - Bp8CacheImpl is a universal container for the cacheable objects. Bp8CacheImpl has the following methods to transfer objects in and out the cache: setCacheltem and getCacheltem.
 - Bp8CacheImpl uses Hash Map to store Bp8CacheImpl objects. getCacheltem and setCacheltem methods are synchronized to ensure correct behavior in a multi threaded environment.
 - To prevent duplicate items, method setCacheltem calls the getCacheltem before creating a key for a new item.
 - The Metadata Cache is enabled or disabled via the following setting in the BPF Explorer administrative tool: Enable Metadata Cache (the default setting is True).

NOTE All metadata except for user accounts and dynamic pick lists is cached.

Inbasket Case count

Goal

Inbasket Case Count is displayed in the BPF Web Application next to the Inbasket name: Inbox (4). The BPF Web Application fetches the Case count only upon initial main page load or for an individual Inbasket upon Inbasket refresh. This functionality is enabled or disabled at the security profile level.

Implementation

The MetadataStore class method getInbasketsDOM0() reads the Case count for each Inbasket. The Inbasket Case counts are included with the Inbasket Case list XML object. The Case count works on the work object level only. Counting on the Case level requires successive queries against both CE and PE with filtering and seems impractical. Getting the Case count when Inbasket is refreshed will not cause any additional server loading.

An option to enable the Case Count feature is in the security profile settings in BPF Explorer. The default setting is set to On for all profiles.

The Case Count call is not executed when:

- Opening a Case from BPF search
- The user is working in a GetNext mode where the Inbasket list is hidden

Database access provider

BPF Business Services utilizes JDBC for BPF Metastore and PE database access. JDBC connection pooling is used when BPF Business Services runs within the Web container.

BPF Business Services utilizes two JDBC data sources: Bp8MetadataDS to the BPF Metastore database and Bp8ProcessDS to the PE database. The data source names and the database type are configurable and defined during the installation procedure. This release supports any vendor-supplied non-XA JDBC drivers.

SQL queries

All SQL statements that are executed against the BPF Metastore are pre-defined and stored in the `Bp8sql.xml` file, which is distributed and deployed as part of the BPF deployment package (`bp8-resources.jar` file).

P8 CE API is used for executing SQL statements against an object store repository.

BPF Business Services Java API

In the present release, the BPF Business Services does not provide the core API facets that could be exposed externally. The core API is the API layer that most directly exposes Case objects, their behaviors, their properties, and the object model assumptions that relate them to one another. The BPF Business Services implementation in this release exposes its functionality strictly for use by the core development and support teams.

Although the BPF Business Services is distributed with Javadocs™, the purpose of this documentation is to aid in the code maintenance and not to document and provide interfaces for external consumption.

BPF Web Application

Architecture

The BPF Web Application is a set of JSP pages and servlets that extend the FileNet P8 Platform Web Application Toolkit and pass BPF Web client requests to the BPF Business Services.

BPF Web client validations and rich browser user interface are implemented using DHTML, HTC behaviors, and JavaScript functions. This functionality was optimized for expedient data entry.

XMLHTTP is utilized as a remote scripting protocol and is encapsulated in a set of functions. Internet Explorer performs XSL transformation on the client workstation. All tasks related to generation and parsing of the XML data streams is performed via the XML DOM objects to minimize the chance of generating a malformed XML. The XMLHTTP module will specify the lowest version number, for the XML parser, that is required for processing when performing the XSLT transformation.

BPF Web Application architecture was designed with the key focus on the following:

- Extensibility
- Scalability
- Reuse of P8 Workplace and Web Application Toolkit provided functionality
- Support for unique BPF Web client architecture

Workplace integration

The BPF Web Application utilizes a number features exposed by Workplace via the Integration Servlet. These features include:

- Object Viewer (Since FileNet P8 Platform Integration Servlet does not expose the Object Viewer, BPF adds a custom function to the Integration Servlet to allow access to the Viewer from the external applications [page name].)
- Case search
- Attach document tool
- Browse folder content (When a folder is attached to a Case, the Workplace Integration Servlet allows browsing of folder content and performs any document-related actions including check-in, check-out, view, etc.)

The following configuration files are used to configure the Workplace integration:

- Bp8ExtTasks.xml
- Bp8ExtCommands.xml

Session management

The BPF Web Application was designed to allow users to continue working without losing data when their web session expires. When a user makes a request to the Web server and the user's session is expired, the Web client will prompt the user to re-enter their credentials, which will be validated by the BPF Web Application, and execute the requested action.

Action Dispatcher

The Action Dispatcher servlet is an XML-based service that accepts requests from the BPF Web client, passes them to the BPF Business Services, packages the responses from the BPF Business Services, and passes them back to the BPF Web client.

IsDirty

Goal

To prevent writing data to the repositories unnecessarily.

When a Case or any associated objects are updated with the changed values, both client and server can use the information to avoid writing to the database when underlying data has not changed. The obvious benefit is to eliminate performance overhead that is associated with the write operation in the first place. There is also a hidden benefit when performing an audit on the field level: the database is spared the overhead of storing huge quantities of unnecessary change tracking. Since in an active database the volume of audit trail information can far exceed the actual volume of the data, the performance and storage benefits of this feature can be significant.

Implementation

The Web client tracks those fields the user, custom script, or a plug-in changed. Only those fields that have changed are posted to the server as part of the Case XML. A Case field is considered changed only if a field value changed from the initial value. That is, if a user opens a Case and changes the Approved Amount field value from 100.00 to 102.12 and changes it back to 100, it is not considered a change and the field is not considered dirty. The old and the new values are compared only after formatting. The following values are considered identical:

MONEY: 100 and 100.00

Date: 01/01/01 and 01/01/2001

The Web Client Java script event handler allows resetting of the Dirty flag via programmatic calls from the custom script. The original value is tracked via the Value attribute of the field; the new value is tracked via the NewValue attribute of the field element.

Since the Web client keeps track of what fields have changed and posts the data stream with the changes only, BPF Business Services saves the field values to the corresponding objects (Case object, attachments, work item, etc) as is. However, the following special circumstances apply:

- Instantiation rule – The objects in the corresponding repository are instantiated and updated only if there were field updates to this object:
 - Document objects (attachments) are not instantiated and updated if there were no field updates to document objects.
 - The workflow (VWObject) object is instantiated anyway since the work object has to be unlocked, saved, or dispatched, so this rule will not apply.
 - The Case object may not need to be instantiated and updated if there were no field changes.
 - Operations associated with the response will always override the data posted by the client. Note that the operations will override the instantiation rule and update the corresponding object if need be.

- The Refresh or Case Open Java plug-in supports handling the dirty fields. That is, this plug-in may set the fields as dirty before the Case XML is set to the user browser.
- The Validation or Case Response plug-in may set the fields as dirty after the user uploads the Case XML.

Lookup field

The Lookup is a special Case field attribute that serves to:

- Provide better performance on pick list fields where the pick list contains a large number of entries (usually in the hundreds range).
- Allow implementing a type ahead pick list. Allow assigning a custom URL as a data source for the field.

Implementation

- The design provides a standard interface to a field associated with a pick list to allow the user to type into a text box instead of downloading the entire list down. For example, on Country pick list, when the user types "Ca" it should return a list of countries starting with "Ca" and when the user types in "Canad" the value of Canada should be populated.
- The lookup pick list searches for an exact value match first and performs a secondary automated wildcard search if lookup for the exact value yields no results.
- There is a generic lookup function working against any Bp8 pick list.
- There is a standard interface to a URL data provider that can call a configured URL with the configured list of fields and their values and be able to receive an interpreted XML message with the result set.

Response user lookup

When the Lookup URL check box is selected in the BPF Explorer Inbasket properties page on the Responses tab, the user assignment prompt is presented as a text box instead of the combo box.

Case field lookup

Standard Case field lookup data entry should be configured as a URL in the BPF Explorer Case field property Lookup URL and be overridden as necessary.

1. When the user types into a field and tabs out (loses focus) or clicks the button next to the lookup field text box, the pick list should execute.
 5. If more than one record is found, display a choice dialog box with the entries that match the request (limit 50) and allow user to select the entry.
 6. If only one record is found, populate the field and follow the control.
 7. If no records are found, display "No matching record found."
2. If no valid entry is made, prevent the user from leaving the field unless a valid entry is made or the field is cleared.

NOTE Typically, field lookup is initiated (triggered) when the user leaves a field. If the Disable Lookup Validation checkbox is checked, the lookup will be triggered by the lookup button instead. A use case example of this would be when you wish to have the user enter multiple fields of entry before invoking the lookup (like entering a multi-field address before looking up the zip code).

Static pick list

The dialog box displayed upon more than a single entry found should include the code and description. A typical example is a Country list containing short and long country names.

Dynamic pick list

The dialog box displayed upon more than a single entry found should include the code and description and any other additional columns defined in the pick list.

A typical example is a user pick list, which also returns custom user attributes, department information, etc.

External URL as a data source

Any external URL (JSP, ASP, etc) could be associated with the lookup field instead of the standard handler. The URL provider should be able to accept parameters in the URL and be able to respond with a standard XML file. BPF should be able to interpret the XML file, and based on the content:

8. Populate one or more Case fields if one record is returned
9. Automatically show a dialog box displaying the data returned in XML
10. Automatically display the "No matching record found" message.

The URL should allow configuring which Case fields will be passed in the HTTP POST call.

One of the typical examples is providing Postal Code connector, where the user can type the postal code and possibly part of the address, such as a street name, and clicking a button next to the lookup field to invoke a custom URL, which would accept the user entry, make a call to a postal database, and send the result back using a standard XML message. The client will automatically interpret the result and populate all the address related fields.

BPF Web client

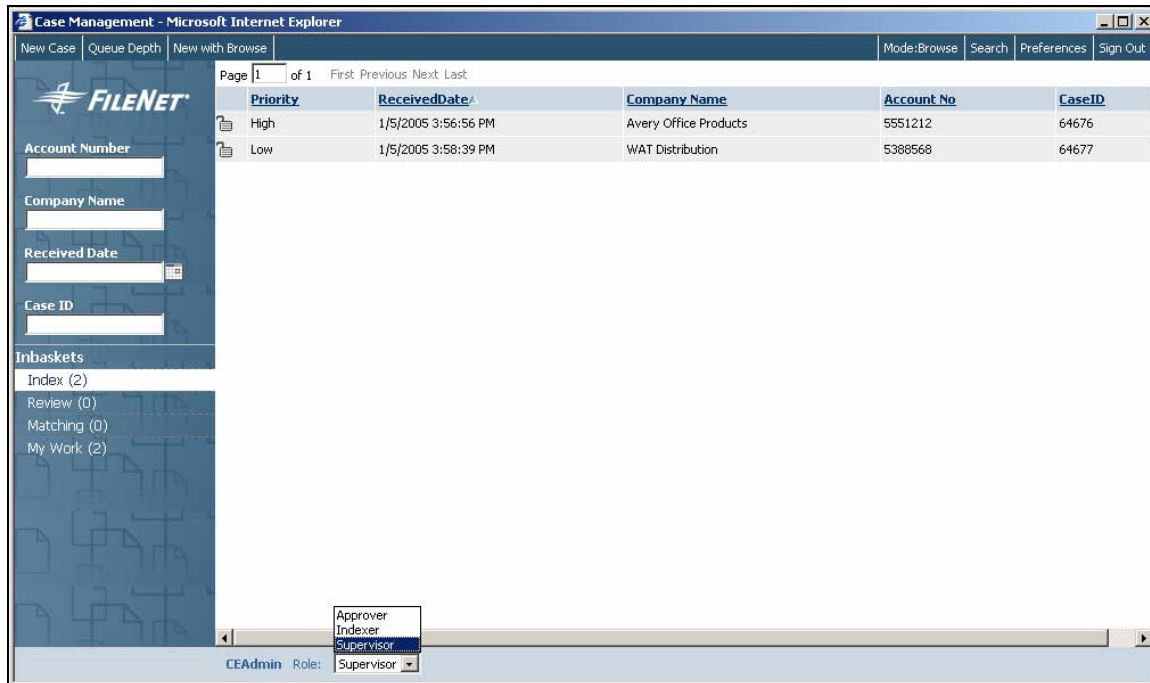
Sign In

The Sign In page is implemented using the P8 Web Application Toolkit. The user, password, and role (optional) fields are available for entry. The following algorithm is used:

Authenticate user in CE (LDAP authentication)

11. Authenticate user in PE (workflow router)
12. If the workflow user ID field is empty in the Bp8 database, it will be retrieved from the PE and updated to the user record in Bp8.
13. In case authentication fails at any of the above steps, all connections already established will be released.
14. Retrieve Workplace access roles list the user has been assigned to and map to BPF security profiles.
15. Determine default security profile and display user interface pertaining to that profile.

Main Page - BPF Web Application



Inbasket List

The following algorithm will be used to allow paging through the list of cases for the end user:

- When the list of cases is returned to the Inbasket, the list of all Case IDs is returned to the browser together with the first page of data. The list of cases will be packed into a compact data structure to allow handling of a large number of Cases in the Inbasket (5000-10000). The structure will be ordered by the current sort order. If the user changes the sort order, the information is reloaded. The estimated size of this structure, even for a large number of cases, was estimated to be comparable to the size of HTML that is passed today for the Inbasket browser by BPF. The maximum number of records to retrieve is configurable. BPF has a configurable limit of the maximum number of cases in a queue.
- The browser displays paging buttons. Handlers for these buttons determine the proper range in this structure and pass it back to the server. The server retrieves browse data for the specified Case IDs and returns it to the client.

Case User Interface

The screenshot displays the IBM FileNet Case Management web application running in Microsoft Internet Explorer. The main content area is titled 'Case Information' and contains several input fields for case metadata. The left sidebar shows a navigation menu with 'Inbaskets' and a list of items: 'Index (2)', 'Review (0)', 'Matching (0)', and 'My Work (2)'. The bottom of the page features a status bar with the user's role ('Supervisor') and the current case ID ('64676').

Field	Value
Case ID	64676
Account Number	5551212
Company	Avery Office Products
Designated Rep	
Effective Date	1/31/2005
Contract Amount	\$21,000.00
Received Date	1/5/2005 3:56:56 PM
Priority	High
Contract Notes	
Document Type	Primary

Concepts

Standard tabs

Case metadata is presented to a user in a tabular interface. Two tabs are provided by default: Case and Audit, which can be configured to be exposed on any Inbasket via BPF Explorer. Custom tabs can also be added to extend the Case model, presenting data from CE or other repositories with the required layout.

Plug-in tabs

The Attachments tab is provided as an example of a plug-in tab. This tab is used to display Case attachments in an HTML table. See the Advanced Configuration documentation for more information.

The Generic Table tab is also provided as a foundation of a plug-in tab when you need a multi-line input form. See the Advanced Configuration documentation for more information.

Tools

Create Case (select a local file)

Purpose

Allows creation of new Case objects from the Web Application. The Case fields list is displayed per the Case fields configured for this tool. If there is more than one Case type configured, the user will be presented with a combo box allowing Case type selection.

Enabling or disabling the entry of a local file (that will be stored in CE upon Case creation) is configured when defining a Case type. The document class for the attachment, the workflow name, and the Case object class name are also configured when defining a Case type. The document attachment will be indexed upon checking into the CE repository according to the Case fields exposed on this tool.

Configuration

Case Type:	Applicable
Tool Name:	create_case
Display Label:	New Case
Handler URL:	Bp8CreateCase.jsp
Window Width:	420
Window Height:	300
Resizable:	Y
Modal:	Y
Visibility:	Browse mode

Create Case (browse for a document or folder)

Purpose

Allows creation of Case objects from the Web Application. Case fields list is displayed as per Case fields configured for this tool. If there is more than one Case type configured, users will be presented with a combo box allowing Case type selection. This tool allows selecting an existing document or folder as an attachment to the Case. User can browse the CE repository or execute a stored search template or My Search.

Configuration

Case Type:	Applicable.
Tool Name:	create_browse
Display Label:	New Case with Browse
Handler URL:	CreateCase.jsp
Window Width:	300
Window Height:	300
Resizable:	Y
Modal:	Y
Visibility:	Browse mode

Add Document

Purpose

Allows attaching documents from the local disk to Case objects Case fields list is displayed as per Case fields configured for this tool. If there is more than one Case type configured, user will be presented with a combo box allowing Case type selection. Document class for attachment is configured on the Case type level. Document attachment will be indexed upon checking in the repository according to the Case fields exposed on this tool.

Configuration

Case Type:	Applicable
Tool Name:	add_document
Display Label:	Add Document
Handler URL:	Bp8AddDocument.jsp
Window Width:	350
Window Height:	300
Resizable:	Y
Modal:	Y
Visibility:	Browse mode

Attach Document or Folder

Purpose

This tool allows attaching existing CE repository documents or folders to the Case. User can browse the CE repository or execute Stored Search template or My Search.

Configuration

Case Type:	Not Applicable
Tool Name:	wp_add_attach
Display Label:	Attach Document
Handler URL:	/ToolRedirect.jsp
Window Width:	640
Window Height:	520
Resizable:	Y
Modal:	N
Visibility:	Case mode

Queue count

Purpose

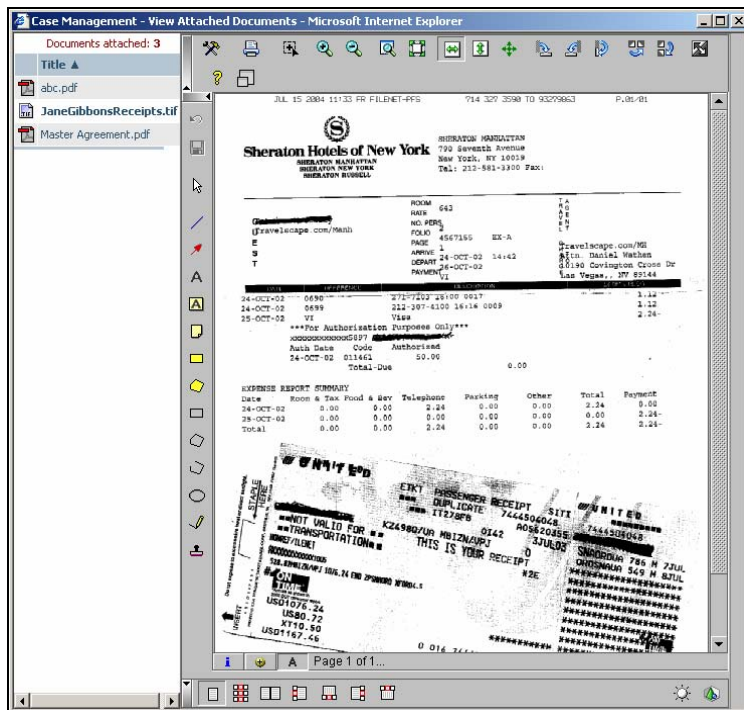
Displays number of work items (queue count) for each Inbasket in the current profile.

Configuration

Case Type:	Not Applicable
Tool Name:	queue_depth
Display Label:	Queue Depth
Handler URL:	Bp8QueueDepth.jsp
Window Width:	300
Window Height:	500
Resizable:	Y
Modal:	Y
Visibility:	Browse mode, Case mode

Object Viewer

Document Viewer

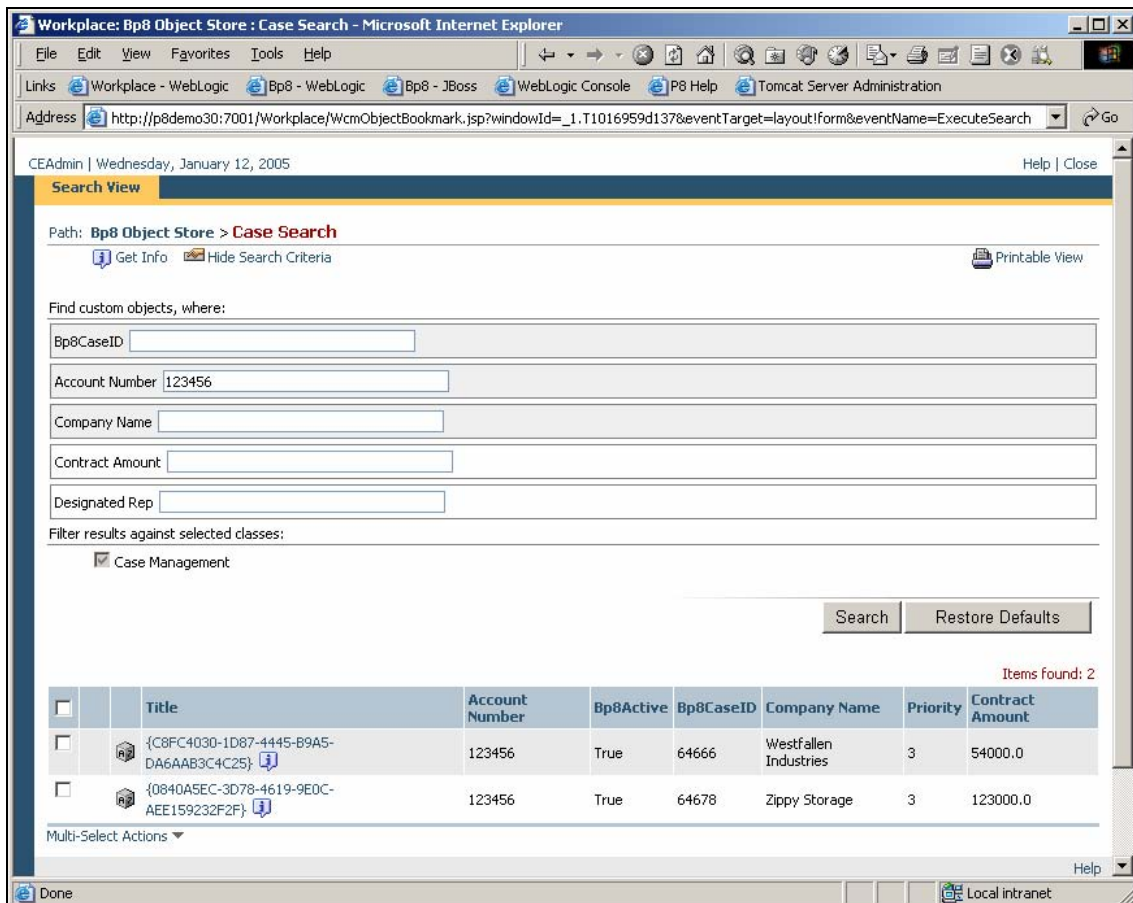
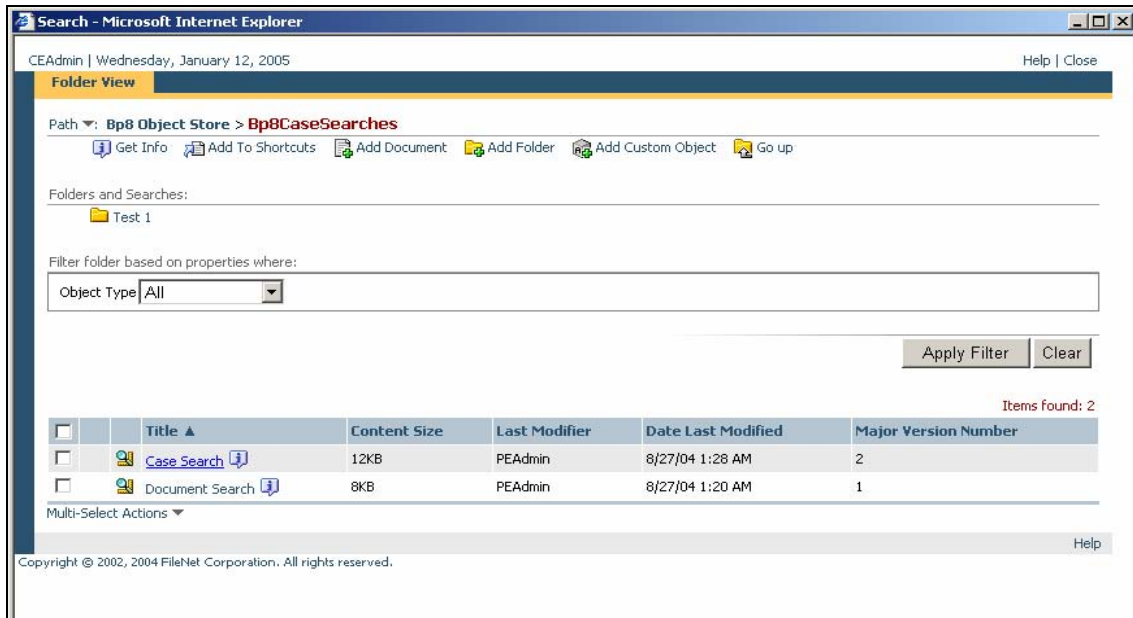


The BPF Document Viewer consists of two main areas: the attachments pane and the document pane. The list of attachments shows all documents attached to the Case. The document page contains the Viewer appropriate for the content being displayed.

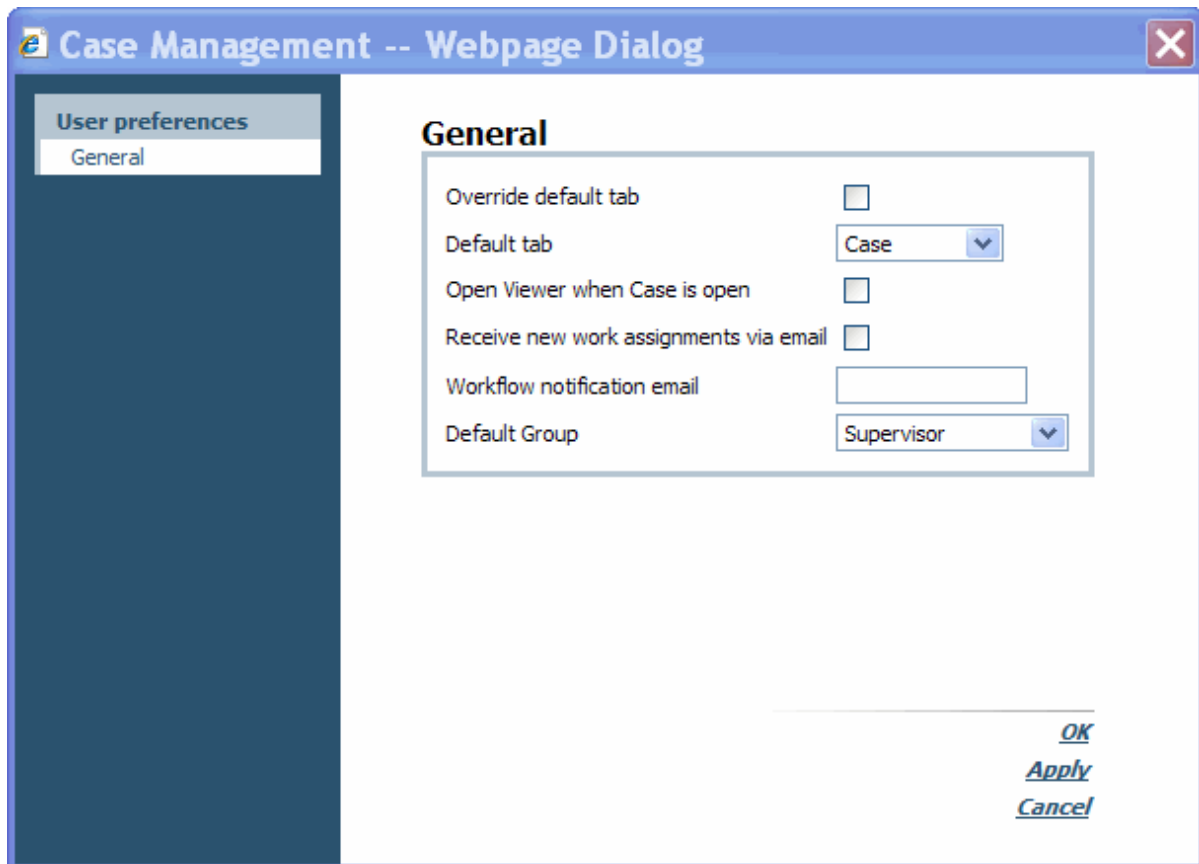
The Document Viewer is invoked via the P8 Workplace IntegrationServlet and is hosted within the Workplace.

BPF Search

Search is executed by clicking the Search button on the Main page tool bar.



User preferences



Case Management -- Webpage Dialog

User preferences

General

Override default tab ☐

Default tab Case

Open Viewer when Case is open ☐

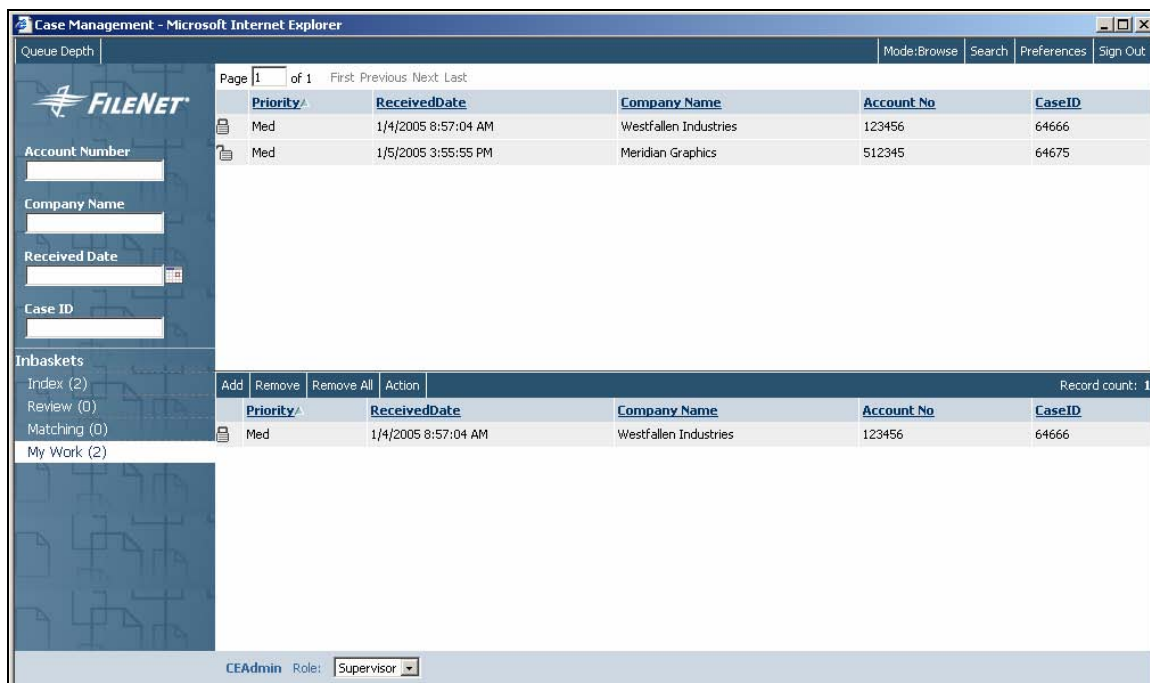
Receive new work assignments via email ☐

Workflow notification email

Default Group Supervisor

[OK](#)
[Apply](#)
[Cancel](#)

Bulk processing



Case Management - Microsoft Internet Explorer

Queue Depth: 1 of 1 [First](#) [Previous](#) [Next](#) [Last](#) [Mode:Browse](#) [Search](#) [Preferences](#) [Sign Out](#)

FILENET

Account Number

Company Name

Received Date

Case ID

Inbaskets

[Index \(2\)](#) [Review \(0\)](#) [Matching \(0\)](#) [My Work \(2\)](#)

Priority	ReceivedDate	Company Name	Account No	CaseID
Med	1/4/2005 8:57:04 AM	Westfallen Industries	123456	64666
Med	1/5/2005 3:55:55 PM	Meridian Graphics	512345	64675

[Add](#) [Remove](#) [Remove All](#) [Action](#) Record count: 1

CEAdmin Role: Supervisor

Sign Out



Localization

The BPF Web Application supports the UTF-8 character set and is designed in such a way that all the metadata can be easily localized via a set of resource files. A single Web Application can support an unlimited number of languages simultaneously. The localized metadata is cached to achieve better performance.

The user selects the default language via the Internet Explorer preferences: **Tools > Internet Options > Languages**. All metadata content will be delivered in the language based on user preference assuming this language has the appropriate resource files available. Please see the *IBM FileNet Business Process Framework Localization Guide* for more information.

Certain data elements (dates, monetary fields) are formatted based on the user's desktop local preferences. The data formatting preferences are controlled via the Regional Options in the Windows® control panel.

BPF Business and Client Tiers extensibility

BPF Web Application tools

BPF Web Application tools allow configuring and exposing a set of actions that extend Case functionality and provide additional features and functionality not provided with the core BPF Web Application. Tools are displayed as HTML links in the toolbar and are configured to either be displayed or hidden per each Inbasket. Tools could be configured to be displayed in the Case or browse mode or both.

Developers can build their own tools and expose these tools on the Web interface to extend Case functionality or provide interfaces to any external systems. Bp8 allows configuring a URL that is executed upon user clicking the link and provides management for the window displayed.

Refer to the *IBM FileNet Business Process Framework Developer Guide* for more details.

BPF Web Application tabs

The Case User Interface is presented to a web client user in a tabular interface. Two tabs are provided by default: Case and Audit, which are configured and exposed on any Inbasket via BPF Explorer. Custom tabs could be added to extend the Case model, presenting the data from CE or other data repositories with the required layout.

The Attachments tab is provided as an example of a custom tab. This tab is used to display Case attachments in an HTML table.

The Table tab is also provided as an example and foundation of a plug-in tab used specifically when you have multiple row entry needed. The tab can be used to enter and store any additional information needed by the case.

Refer to the *IBM FileNet Business Process Framework Developer Guide* for more details.

BPF lookup handler

BPF Case fields can be associated with a URL handler, which allows passing this and other Case field values to a URL and accepting results back. The data is packaged as XML and is posted as part of the HTTP POST request to the configured URL. The URL can handle the data in any way required and pass the result back in form of an XML packet containing the following:

- A user message
- Set of Case fields and their values to be populated back to the Case
- A group set of Case fields records and their values to be populated back to the Case. The BPF Web Application will display an interface to allow the user selecting the desired group set.

A Lookup URL handler allows validating the field data against an external data source or a rules engine.

Refer to the *IBM FileNet Business Process Framework Developer Guide* for more details.

BPF Web Client JavaScript API

Goal

Provide web client interfaces for developers to extend and control the web client behaviors.

BPF allows extending the BPF Web Application behavior with custom JavaScript functionality executed when the user opens the Case, manipulates the Case, or completes the Case.

Implementation

The BPF Web Application main page event handler interface is implemented as `EventHandler.js` JavaScript include file and is deployed in the `plug-ins/custom/` folder. The `EventHandler.js` file provided with the base BPF package contains stubs to all supported event handler functions.

BPF Business Services Java adaptor interfaces

Goal

To allow extending domain object model with data from external system and provide an EAI layer.

The BPF Business Services model allows for virtual extensions and custom functionality by associating custom Java classes (adaptors) to the Get Case and Case Complete events. Java adapters implement standard interfaces that receive Case object as input, and return Case object back (presumably after data manipulation).

- The `getCase` adaptor is associated with the Case Open event and is pluggable in the BPF Web Application via the Inbasket configuration.
- The `doFunction` adaptor is associated with the Case Complete and is pluggable in the BPF Web Application via an Inbasket configuration.
- BPF defines standard interfaces for the `getCase` and `doFunction` adaptors.

The Case is presented to the `getCase` adaptor consumer as a Java object that is a collection of properties; serialization and de-serialization of the Case XML DOM object is a responsibility of the BPF Business Services.

Interactions

getCase Java Adaptor

If a getCase Java Adaptor is configured for an Inbasket that the Case is being opened from, the configured Adaptor will receive the Case object properties. The Adaptor will execute the custom business logic, modify the Case data as necessary, and return the control back to the BPF Business Services. BPF Business Services will complete the operation. The doFunction Adaptor will return an indication of success, warning, or failure, with an informational message in case of warning, or an error message in case of failure. The BPF Web Application processes warning and error level messages. Warning level raises a user message allowing canceling or continuing the operation, while error level raises the message and aborts the operation.

doFunction Java Adaptor

If a doFunction Java Adaptor is configured for an Action that was requested by the client, the configured Adaptor will receive the Case object properties. The Adaptor will execute the custom business logic, modify the Case data as necessary, and return the control back to the BPF Business Services. BPF Business Services will complete the operation. The doFunction Adaptor will return an indication of success, warning, or failure, with an informational message in case of warning, or an error message in case of failure. The BPF Web Application processes warning and error level messages. Warning level raises a user message allowing canceling or continuing the operation, while error level raises the message and aborts the operation.

NOTES

- The doFunction adaptor name, a Java class name, is configurable on the Inbasket Response level (**BPF Explorer > Inbasket Properties > Responses tab > Java class** for the selected response).
- Each Inbasket can have multiple doFunction adaptors, but only one per each Inbasket Response.
- The getCase adaptor name (Java class name) configurable on the Inbasket level (**BPF Explorer > Inbasket Properties > General tab > Java class**).

BPF Web Application user preferences

The User Preferences module implements BPF user preferences and allows defining and extending the preferences structure. User preference values are stored in CE custom objects.

This release supports the following attribute types: String, Integer, Boolean, and Option. New types could be defined as plug-ins.

BPF Web Client user interface customizations

XSL transformation

Most of the BPF Web Application user interface elements are rendered to HTML via an XSL transformation; the data itself is in the presentation-neutral XML. It is possible to further customize the user interface by modifying the XSL modules, optionally associating custom JavaScript to customize the presentation even further. The following list defines the major XSL files:

Bp8CaseProps.xsl	Case tab
Bp8CaseFunctions.xsl	Case functions
Bp8CaseAttachments.xsl	Case attachments list
Bp8CaseAudit.xsl	Audit tab
Bp8Cases.xsl	Inbasket Cases list
Bp8Inbaskets.xsl	Inbaskets list
Bp8User.xsl	User context
Bp8Filters.xsl	Inbasket filters

Images and style sheet

HTML styles, background images, and colors are stored in the CSS style sheet and could be customized to tailor the look and feel to the customer's requirements.

Font sizes, colors, background images

Edit \css\Bp8Styles.css

Logo in the Inbasket area

Replace \img\Logo.gif with your image

Splash image

Replace \img\Logon_logo.gif with your image

Logon page

Replace \img\p8RightLogOnBar.jpg with your image

Replace \img\p8LeftLogOnBar.jpg with your image

Background image in the Inbasket area

Replace \img\BannerElements.jpg with your image

BPF configuration

Inbaskets

- Name
- Security Profile
- Work filtering
- Work delivery mode
- Work list configuration
- Actions
- Attachments list
 - Visibility
 - SQL statement
- Logging the Case Open event
- Tabs visibility and order
- Tools visibility and order
- Java adapter

Workflow Queue Filtering

- Pre-defined workflow queue filters
- Queue name
- One or multiple workflow steps

Inbasket Filtering

- Inbasket filters visibility and order
- User-initiated Inbasket content filtering

Work Delivery

- Enforced Pull mode
- Enforced Push mode
- Or user selectable

Case List

- Configurable column list
 - Column name
 - Case field associated with the column
 - Default sort order and direction
 - Pick list code or description
 - Column could be defined as sort-able
- Default sort order
- Paging support
- Bulk processing

Responses

- Configurable labels for workflow responses
- List is dynamic and based on the workflow step definition
- Configurable hot keys
- Configurable actions order
- Configurable required fields

Operations

Operations allow assigning a value to the selected Case field while the response is being executed. Operations will override any field modifications that user has made. There could be multiple operations associated with a response. Each operation is tied to a single Case field, is based on the built-in (macro) selected, and operates within the context of the Case and the user.

Built-in	Value	Description
Case field	<Case Field Name>	A name of another Case field. This built-in allows assigning a Case field value to a Case field. Case field name is configured in Case field properties in BPF Explorer.
Constant		A name of another Case field. This built-in allows assigning a Case field value to a Case field. Case field name is configured in Case field properties in BPF Explorer.
Active User ID	empty	Active user BPF user ID. This is the value from the USERS table, USER_ID column in the BPF Metastore.
Active User PE ID	empty	Active user workflow ID (PE user ID). This built-in is typically used for implementing personal work assignment.
Active User Name	empty	Active user short name (typically a logon name).
Active User Full Name	empty	Active user full name.
Active User Profile Name	empty	Security profile name of the active user.
Active User Profile ID	empty	Security profile ID of the active user.

Case User Interface

Case fields

Configurable features

- Label
- Name
- Type
- Pick list
 - Combo box field
 - Lookup field
- Display width
- Max chars allowed for entry
- Number of rows

Key behaviors

- Data formatting based on the user's desktop regional settings
- Inline data validations based on the field type
 - Invalid data error message suggests the valid format for the selected locale

Tabs

- Case tab
 - Layout is configurable via a graphical editor
 - Collapsible sections (expandos)
- Audit tab
- Attachments tab
- Table tab

Tools

- Add new document
- Attach repository document or folder
- Create new Case with selecting an existing document or folder or browsing for a local file
- Inbasket queue counts
- Configurable reason lists
- Configurable required fields

Misc

- Inline data validations

BPF Search

- Utilizes CE Stored search templates
- Case is presented in the user interface matching the Inbasket presentation
 - Configurable on the security profile level
 - Could be read-write or read-only
 - Tools could be exposed
 - No work object locking
 - Inbasket name and bound user name is displayed

Globalization and localization

This section describes globalization and localization support in BPF.

Globalization

The globalization of BPF is implemented by separating localizable messages (translated text) into resource files. All BPF components utilize the same set of resource files. Each resource file is separated into core and custom parts. The core resource file implements the generic settings, such as error messages and BPF Web Application user interface labels, which do not change in the custom implementations. The custom resource file implements the custom elements, such as Inbasket names, Case field label names, etc, which vary implementation to implementation.

Each line in the properties file is a key=value pair. The main advantage is that file content can be easily organized and modified without recompilation.

Localization

Localization is accomplished by translating the text in a resource file to the desired languages.

A single Web Application can support an unlimited number of languages simultaneously.

Users select their default language via the Internet Explorer preferences: **Tools > Internet Options > Languages**. All metadata content will be delivered in the language of the user's choice assuming this language has a resource defined.

All localization resources are stored as text files within the `bp8-resources.jar` file located in the `lib\` folder in the deployed Web Application. All localization resources for each specific country and (or) language are located in a separate file. The name of the file contains both language and country names as follows:

```
strings.ll_CC.txt
```

where:

ll - two-letter language code in lowercase, for example:

en, fr, de, ru.

CC - two-letter country code in uppercase, for example:

US, FR, DE, RU.

Example: `strings.en_US.txt` for US English

Country is optional and can be omitted, for example: `strings.en.txt`.

Format:

Example:

^Key ^Value

Metadata localization

- All static content of the BPF Web Application pages
- Error messages
- Case field definition
 - Display name
- Inbasket Definition
 - Display name
- Security profile definition
 - Display name
- Pick list definition
 - Data

BPF Web Application Case field value formatting is based on the user desktop locale preferences. The data formatting preferences are managed via the Regional Options applet in the Windows control panel.

- **Numeric** – Based on the numeric preferences in the Regional Options. This setting defines the decimal symbol, digit grouping symbol, etc.
- **Money** – Based on the currency preferences in the Regional Options. Only a single format for the negative currency value is supported where the dash (-) sign is set to display before the value. The dash (-) sign after the value or after the parenthesis [()] sign are not presently supported.
- **Date / Time** – Format is defined based on the date and time elements in the Regional Options. When a year is entered as a double digit instead of four digits, the logic of assigning the proper year is based on the calendar setting in the date section of Regional Options.

Case field values are not localizable. User can enter non-English text in a string property value however the data storage and retrieval is subject to the Content and PE requirements and constraints.

NOTE BPF pick list values are localizable.

BPF Explorer

BPF Explorer is a Microsoft Management Console Snap-in that allows configuring the BPF applications and features.

Architecture

BPF Explorer is built using Microsoft Visual Basic 6.0 and Microsoft MMC ActiveX. BPF Explorer connects directly to the BPF Metastore database utilizing an JDBC driver. Users are required to enter the logon and password to sign on to BPF Explorer. A database user account with the permission to modify the BPF Metastore should be used. All changes to the BPF configuration is immediately reflected in the BPF Metastore database after user confirms the change.

Initial load

The following defines the system registry configuration settings utilized by the BFP Explorer:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\FileNet Business Process Framework]
"DATABASE_CONNECTION_STRING"="DSN=<database server name>;UID=;PWD="
"DATABASE_TYPE"="mssql"
"DATABASE_SQL_XML"="C:\\Program Files\\FileNet Business Process
Framework\\BPFEplorer\\Bp8sql.xml"
"DEFAULT_OBJECTSTORE_NAME"=" "
"LAST_USER_CE"=" "
"LAST_USER_METADATA"=" " "
```

Logon

Managing Multiple BPF Metastores

BPF Explorer allows managing multiple BPF Metastores from the same console. The selection as to which BPF Metastore to manage is made during logon where the user is provided with the list of the registered repositories to choose from.

In order to register a new BPF Metastore, the following data needs to be manually entered into the registry:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\FileNet Business Process Framework\Postcorb]

"DATABASE_CONNECTION_STRING"="DSN= Postcorb;UID=;PWD="

"BP8_EXPLORER_DATABASE_CONNECTION_STRING"="DSN=Postcorb;UID=;PWD="

"DATABASE_TYPE"="mssql "

"DATABASE_SQL_XML"="C:\\Program Files\\FileNet Business Process
Framework\\BPFExplorer\\Bp8sql.xml "

"DEFAULT_OBJECTSTORE_NAME"=" "

"LAST_USER_CE"=" "

"LAST_USER_METADATA"=" "

[HKEY_LOCAL_MACHINE\SOFTWARE\FileNet Business Process Framework\Case Management]

"DATABASE_CONNECTION_STRING"="DSN=CaseManagement;UID=;PWD="

"BP8_EXPLORER_DATABASE_CONNECTION_STRING"="DSN=CaseManagement;UID=;PWD="

"DATABASE_TYPE"="mssql "

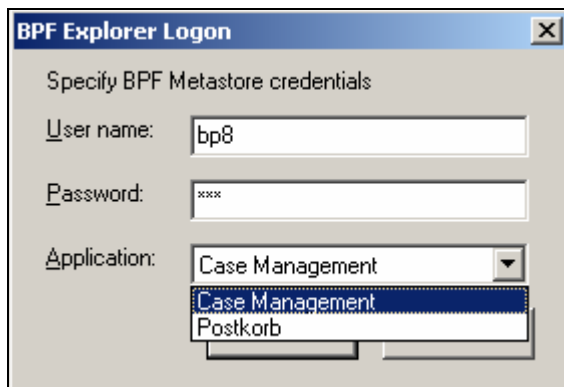
"DATABASE_SQL_XML"="C:\\Program Files\\FileNet Business Process
Framework\\BPFExplorer\\Bp8sql.xml "

"DEFAULT_OBJECTSTORE_NAME"=" "

"LAST_USER_CE"=" "

"LAST_USER_METADATA"=" "
```

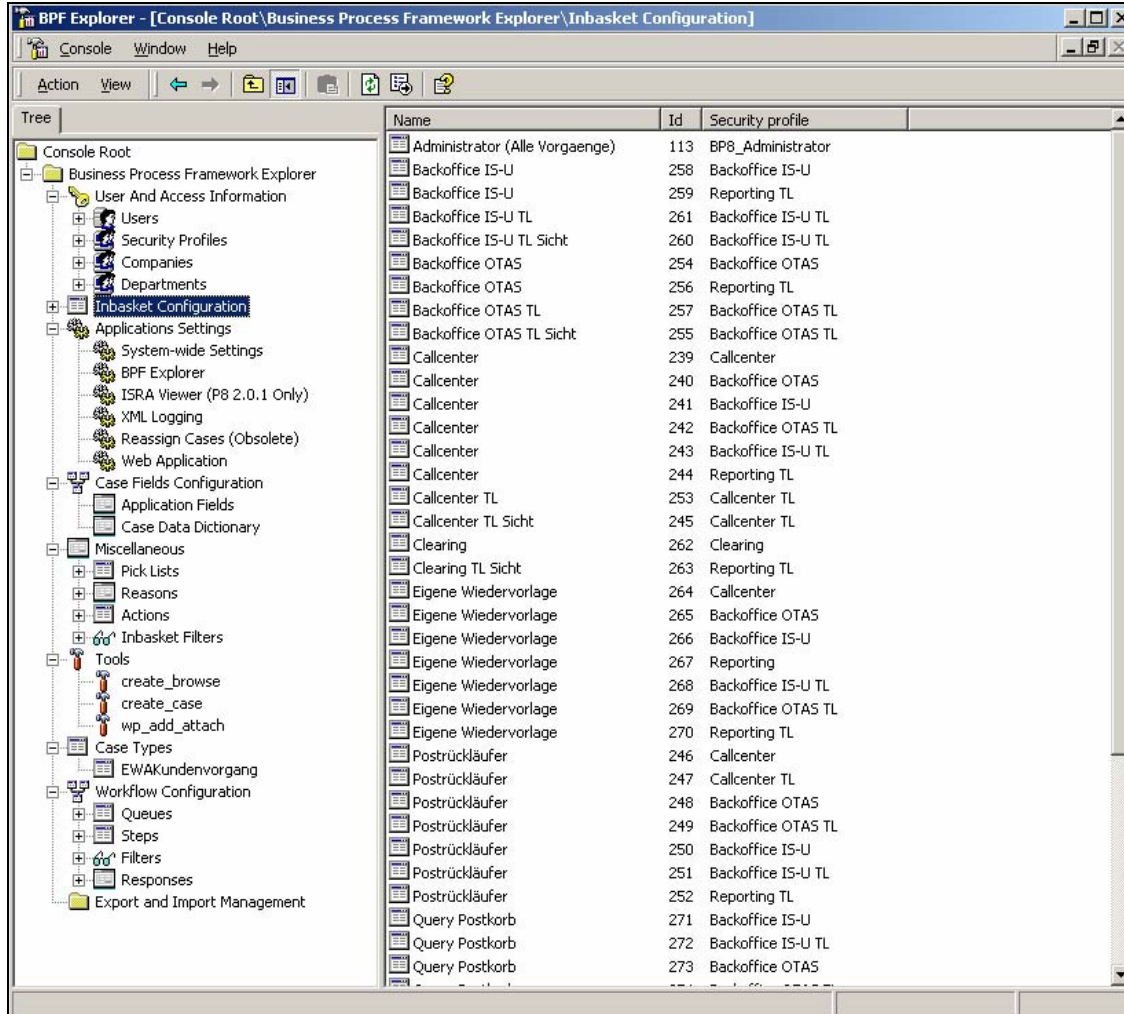
BPF Explorer will display a combo selection box with Postcorb and Case Management in the choice list upon start. Postcorb and CaseManagement data source names need to be manually created and pointed to the appropriate BPF Metastore databases. The BPF Explorer logon window with multiple repository option is shown below.



Features and functionality

Main Page

The BPF Explorer main window is shown below.



Export and import manifest

The Configuration Export/Import tool is part of the BPF Explorer that allows saving the metadata configuration to an XML manifest and importing the configuration from an XML manifest. A BPF manifest could also be generated as a SQL script, which could be initiated against a BPF Metastore database to import the data.

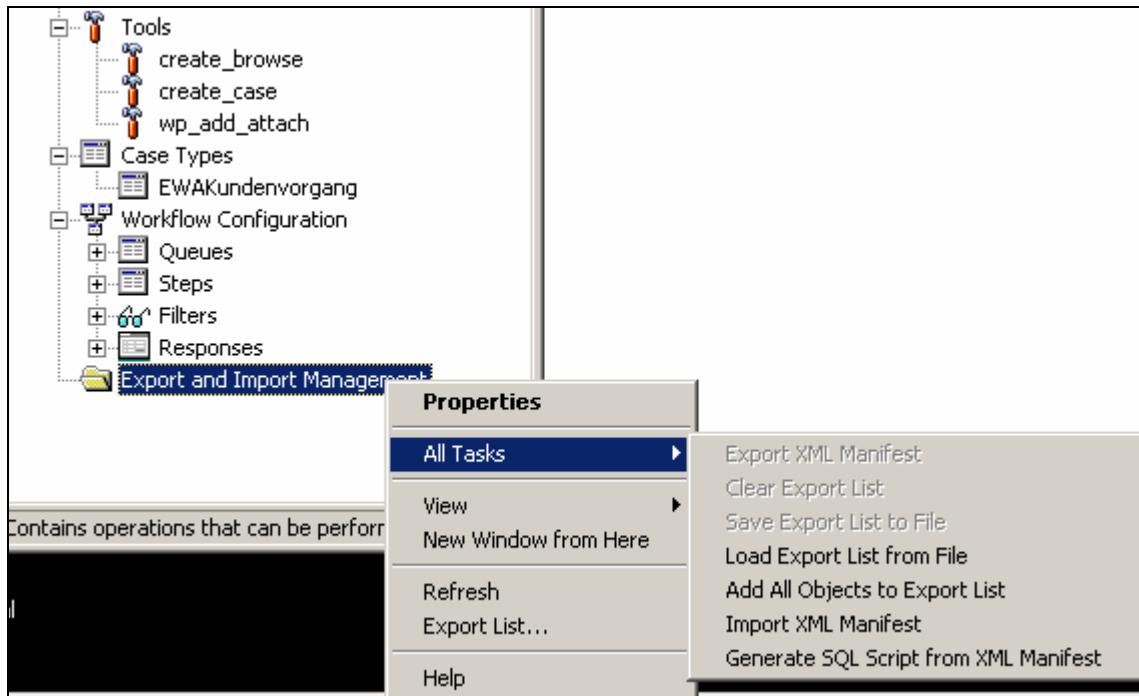
The Export/Import tool allows selecting the individual objects to export (i.e. Inbasket, security profile, etc), all objects of the same type, of the entire configuration. The list of selected objects can be stored in an XML export list file and imported later. This not to be confused with the XML export file, which contains the actual exported data.

The current implementation supports importing a full configuration only.

The layout of the BPF Manifest and all SQL statements being executed upon import and export are configurable via the `ConfExpImp.xml` file.

Export Manifest Node

The BPF Explorer Export and Import context menu is shown below.



- Properties - open the Export Manifest Properties dialog (see below).
- All Tasks
 - **Export XML manifest** – Export all objects, included into the export manifest, to the XML file. The command shows the output configuration file selection dialog.
 - **Clear export list** – Clears the export list.
 - **Save export list to file** – Saves the export manifest itself to the XML file. The command shows the output export manifest file selection dialog.
 - **Load export list from file** – Loads the export manifest itself from the XML file. The command shows the input export manifest file selection dialog.
 - **Add all objects to export list** – Adds all available objects into the export manifest. This is the one way to create a full configuration, which can be imported in the current implementation.
 - **Import XML manifest** – Imports objects from a saved configuration to the current BPF Metastore database. Only full configurations can be loaded. The command shows the input configuration file selection dialog.
 - **Generate SQL script from XML manifest** – Generates a SQL script for importing a configuration from a saved configuration. Only full configurations can be loaded in the current implementation. The command shows the input configuration file and output SQL script file selection dialogs.

Export process progress

The BPF Explorer Export progress window is shown below.



Export Manifest properties

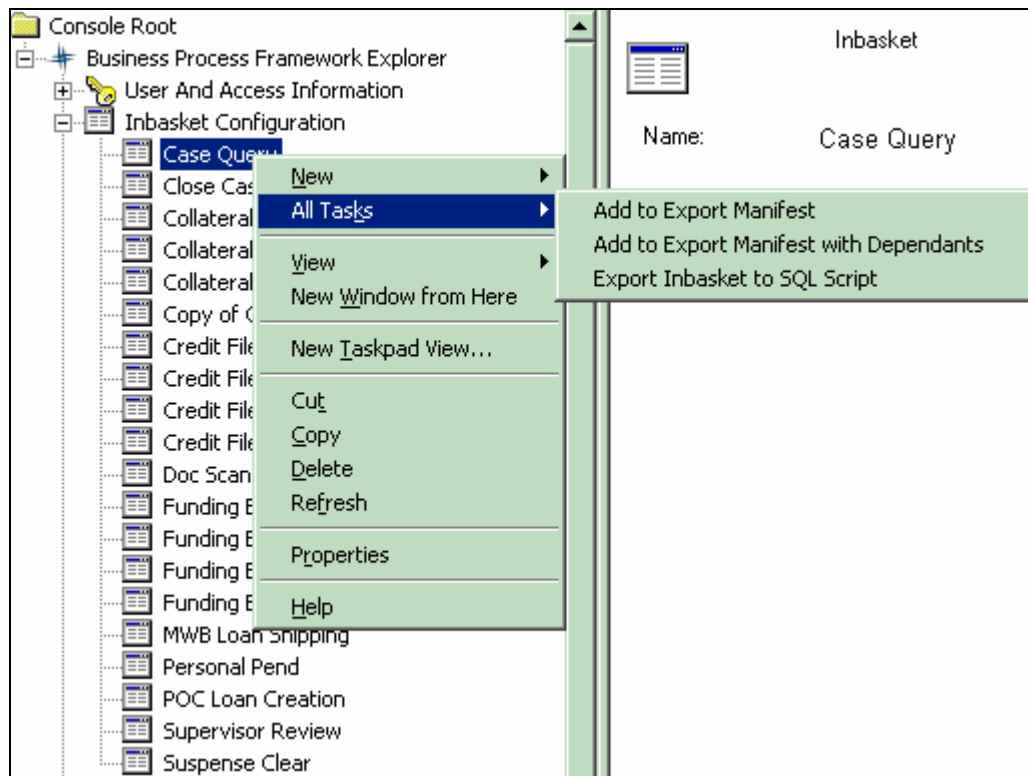
The Export Manifest properties window is designed to add groups of objects to the export manifest (with dependent objects or not), removing objects and groups from the export manifest, adding all objects, and clearing the export manifest.

The Manifest list box, shown on the right, supports sorting by column header all objects and groups that appear in the list box.

The Add With Dependents checkbox is disabled for object types that cannot have dependants (tabs, etc).

Add to Export Manifest

Almost all nodes have the Add To Export Manifest command (i.e. Inbasket configuration node's command adds all Inbaskets into the export manifest). Alternatively, Case query Inbasket node's command adds only this Inbasket. The Add to Export Manifest with Dependants command adds the objects (or group of objects of same type) with all dependent objects (in the case of Inbaskets it is profiles, Case types, application fields, etc.).



The command is available also for objects in the right panel of BPF Explorer:



BPF Operations - Component Integrator Component

BPF provides a BPM Component Integrator component (called BPF Operations) that allows creation and manipulation of a BPF Case object from a Component Integrator step on a process map.

The following table provides a list of the functions exposed via the BPF Operations component.

Architecture

BPF operations are implemented as P8 Component Integrator Java class and provide the following functionality:

- Creating Case objects
- Attaching documents or folders to one or many existing Cases
- Creating audit trail records
- Generating Case ID

Logging

BPF Operations adapter provides log4j based logging.

Create object

This method is used to create BPF Case objects

Flow of events

This use case begins when the workflow map component step is configured with the createObject method to be executed.

16. Create a custom case object and populate its property names
17. Attach an instance of the object to the specified `Bp8Attachment` workflow field.
18. The Component Integrator dispatches the work item to the next step and the use case ends.

Alternative paths

- Invalid object store name
- Required property not populated

Special considerations

Syntax

```
createObject(  
String objectStoreName,  
String objectClassName,  
String objectAttachmentFieldName,  
String[] objectFields)
```

Parameters

{PRIVATE} Name	Required	Default Value	Type
objectStoreName	Yes	-	String
objectClassName	Yes	-	String
ObjectAttachmentFieldName	Yes	-	String
ObjectFields	No	-	String[]

Parameters description

- **objectStoreName** – Name of the CE object store where the instance object should be created.
- **objectClassName** – Name of the CE class used to create the object.
- **objectAttachmentFieldName** – Name of the `Bp8Attachment` workflow field the newly created object should be attached to.
- **objectFields** – Array of field mapping for the object being created. Each entry consists of the following elements:
 - CE object symbolic field name
 - Field type. Valid values are: DATETIME, NUMERIC, STRING, BOOLEAN, MONEY
 - Value could be static value or workflow field or expression

Examples:

```
{"LoanNumber", "STRING", "1234567890"}
```

```
{"LoanAmount", "MONEY", "LoanAmount"}
```

Create Case

This method is used to create an instance of the `Bp8Case` object.

Flow of events

This use case begins when the workflow map component step is configured with the `createCase` method to be executed.

Obtains next available Case ID.

Creates a Case object and populates the property names.

Creates an audit log object indicating that the Case was created.

Attaches documents to the Case if applicable.

Attaches a Case object to the `Bp8Case` workflow field.

Populates the `Bp8CaseID` workflow field with the Case ID.

The Component Integrator dispatches work item to the next step, the Case object is created, and the use case ends.

Alternative paths

- Invalid object store Name
- Required property not populated

Special considerations

- Steps 3 - 5 should be executed in batch mode

Syntax

```
createCase(java.lang.String objectStoreName, java.lang.String caseClassName,
filenet.vw.api.VWAttachment[] attachments, java.lang.String actionName, java.lang.String
actionUserName, java.lang.String[] caseFields)
```

Parameters

{PRIVATE} Name	Required	Default Value	Type
objectStoreName	Yes	-	String
caseClassName	Yes	-	String
caseAttachmentFieldName	No	"Bp8Case"	String
attachment	No	-	VWAttachment[]
actionName	No	"Case Created"	String
actionUserName	No	<CI user name>	String
caseTypeID	Yes	-	String
caseFields	No	-	String[]

Parameters description

- **objectStoreName** – Name of the CE object store where the Case will be created.
- **caseClassName** – Name of the CE class used to create the Case object. The class should be inherited from the `Bp8Case` object.
- **caseAttachmentFieldName** – Name of the `Bp8Case` workflow field the newly created object should be attached to.
- **attachment** – Name of the workflow `Bp8Attachment` property array containing documents. If this property is populated, the component will attach all documents to the newly created Case.
- **actionName** – Name of the action written to audit log when the Case is created. The default value is Case Created and can be overwritten by populating this property.
- **actionUserName** – The user name that will be written to the audit log as part of the event. By default, the name of the interactive user should be written.
- **caseTypeID** – The Case type ID of the Case being created.

- **caseFields** – An array of field mapping for the Case being created. Each entry consists of the following elements:
 - Case object symbolic field name
 - Field type. Valid values are: DATETIME, NUMERIC, STRING, BOOLEAN, MONEY
 - Value could be a static value or a workflow field or expression

Examples:

```
{"LoanNumber", "STRING", "1234567890"}
```

```
{"LoanAmount", "MONEY", "LoanAmount"}
```

Update Case

This method is used to update the CE Case object and its mapped process and attachment fields.

Flow of events

This use case begins when the workflow map component step is configured with the logEvent method to be executed.

Creates an audit log object for the Case identified via the `Bp8Object` workflow attachment field.

The Component Integrator dispatches the work item to the next step, the audit log object is created, and the use case ends.

Alternative Paths

- `Bp8Object` field does not exist on the workflow item
- Required property not populated

Syntax

```
updateCase(java.lang.String objectStoreName, java.lang.String[] caseIDs, java.lang.String[]  
caseFields)
```

Attach document

This method is used to attach existing repository documents to a Bp8 Case.

Flow of events

This use case begins when the workflow map component step is configured with the attachDocument method to be executed.

Executes an adhoc search specified in searchCriteria against the CE repository and locates documents that satisfy the criteria.

For each document object to be attached:

 For each document attached to the attachment property

 Create a new Bp8Attachment object

 Creates an audit log record:

 Action = actionName

 Reason = <empty>

 Description = name of the document being attached

 The Component Integrator then dispatches the work item to the next step, and use case ends.

Alternative Paths

If searchCriteria finds no matching documents, the work item should remain in the queue.

Required property not populated should throw an exception.

Syntax

```
attachDoc(java.lang.String objectStoreName, filenet.vw.api.VWAttachment[] attachments,  
java.lang.String auditAction, java.lang.String userName, java.lang.String sql)
```

Parameters

{PRIVATE} Name	Required	Default Value	Type
attachment	Yes	-	VWAttachment[]
actionName	No	"Document Attached"	String
actionUserName	No	<CI user name>	String
searchCriteria	No	-	String

Property descriptions

- **attachment** – Name of the array of workflow VWAttachment properties containing documents. The documents should be already attached to the work item before references to the attached documents or folders. If attachment field is an array containing multiple documents, all documents in the array will be processed.
- **actionName** – Name of the action written to audit log when the documents are attached. The default value is document attached and could be overwritten by populating this property.
- **actionUserName** – Name that will be written to the audit trail as part of the event. By default, name of the interactive user should be written.
- **searchCriteria** – Search criteria to identify which Cases should the document be attached to.

Example:

```
"SELECT Id FROM LoanPackage WHERE LoanNumber='" + LoanNumber + "'"
```

Log event

This method is used to create new Bp8 Audit Log objects (records) that are associated with an existing Case. The Case object should already be attached to the workflow object as VWAttachment named Bp8Object.

Flow of events

This use case begins when the workflow map component step is configured with the logEvent method to be executed.

Creates an audit log object for the Case identified via the Bp8Object workflow attachment field.

The Component Integrator dispatches the work item to the next step, the audit log object is created, and use case ends.

Alternative Paths

Bp8Object field does not exist on the workflow item.

Required property not populated.

Syntax

```
logEvent(filenet.vw.api.VWAttachment attachment, java.lang.String actionName, java.lang.String  
actionReason, java.lang.String actionDescription, java.lang.String actionUserName, int  
eventCategory)
```

Parameters

{PRIVATE} Name	Required	Default Value	Type
caseObject	Yes	-	CustomObject
actionName	Yes	-	String
actionReason	No	"Bp8Case"	String
actionDescription	No	"Case Created"	String
eventCategory	Yes	1	int

Property descriptions

- **caseObject** – Case Object reference. Case object is typically attached to the work item as a `Bp8Object VWAttachment` type object.
- **actionName** – Name of the action written to the audit log.
- **actionReason** – Name of the action written to the audit log.
- **actionDescription** – Description of the action written to the audit log.
- **eventCategory** – Should always be set to 1.

Deployment

All BPF J2EE components are deployed via the same ANT script that is utilized in the compile procedure.

Packaging

Package contents

A BPF deployment package consists of the following:

- A set of export files to set up a CE and PE for BPF
- A set of export files to set up a CE and PE for the BPF Case Management Starter Solution
- Documentation
- BPF J2EE applications binary deployment package
- BPF COM applications binary deployment package

CE objects

The BPF Case-related classes need to be imported into a CE object store to enable case management. The BPF package contains two CE XML export files:

- `ce_base.xml` – BPF core classes definition
- `ce_cm.xml` – Objects and properties for the BPF Case management solution template

Workflow region configuration

The BPF class fields, indices, and Component Integrator Adapter definition need to be imported into a PE workflow region. The BPF package contains the following files:

- `Qcreate_base.xml` – BPF core fields, indices, and Component Integrator Adapter definition. This xml file needs to be run via the BPF WFUtil tool.
- `Qcreate_cm.xml` – BPF Case Management solution template queues, fields, indices, and Component Integrator Adapter definition. This xml file needs to be run via the BPF WFUtil tool.
- `CaseManagement.pep` – BPF Case Management solution template workflow map.

BPF configuration

The BPF Metadata Manifest needs to be imported to the BPF Metastore via the BPF Explorer. The manifest contains BPF metadata definition and application settings.

- `Bp8_base.xml` – BPF core objects definition
- `Bp8_cm.xml` – BPF Case Management solution template objects definition

J2EE applications binary package

All BPF J2EE applications are distributed and deployed via an ANT script and include the following:

- ANT distribution used for deploying BPF
- The BPF Web Application
- BPF Component Integrator Java adapter
- BPF workflow region management utility (wfutil)
- BPF encoding utility (util)
- COM Applications Binary Package

All BPF COM applications are distributed via an Install Shield setup and include the following:

- BPF Explorer
- BPF workflow import tool (wfimport)

BPF Metastore SQL

A BPF deployment package contains a set of the SQL scripts used to generate a new BPF Metastore database or upgrade an existing one. These are slightly different based on database type. The steps for each database have been described below:

Oracle database:

- **Step0.oracle.Bp8.Metabase.1.0.sql**
SQL scripts to create a new tablespace and user information in existing Oracle database.
- **Step1oracle.Bp8.Metabase.1.0.sql**
SQL scripts to generate BPF Metastore database structure in Oracle database.
- **Step2.oracle.Bp8.Metabase.1.0.sql**
SQL scripts to generate new or upgrade BPF Metastore application settings.
- **Step3.oracle.Bp8.Metabase.1.0.sql**

SQL scripts to upgrade BPF Metastore database structure.

MS-SQL database:

- **Step1.mssql.Bp8.Metabase.1.0.sql**
SQL scripts to create a new database and user information in SQL Server and will also generate BPF Metastore database structure in database
- **Step2.mssql.Bp8.Metabase.1.0.sql**
SQL scripts to generate new or upgrade BPF Metastore application settings
- **Step3.mssql.Bp8.Metabase.1.0.sql**
SQL scripts to upgrade BPF Metastore database structure

IBM DB2 database:

- **Step0.DB2.Bp8.Metabase.1.0.sql**
SQL scripts to create a new database schema in DB2
- **Step1.DB2.Bp8.Metabase.1.0.sql**
SQL scripts to generate BPF Metastore database structure in DB2 database
- **Step2.DB2.Bp8.Metabase.1.0.sql**
SQL scripts to generate new or upgrade BPF Metastore application settings

Documentation

The following set of the documentation is distributed with the build. The documentation is distributed as a WAR file and could be deployed as a web application.

- *IBM FileNet Business Process Framework Release Notes*
- *IBM FileNet Business Process Framework Installation Guide*
- *IBM FileNet Business Process Framework Developer Guide*
- *IBM FileNet Business Process Framework Explorer Handbook*

Deployment types

ANT based deployment

Using the ANT script for deployment ensures that BPF J2EE applications are deployable on any ANT-supported platform, including Windows and most UNIX® platforms. ANT does not require a graphical console and can be run from any text-based console. The ANT script performs the following when executed manually:

- Extracts the BPF applications from the archived packages
- Updates the BPF applications with the versions of P8 Content and PE Java API installed on the target server
- Configures the BPF Web Application according to the environmental settings
- Deploys the BPF Web Application to the target Web Application container

BPF Web Application needs to be deployed as an entire package only, interim fixes or partial upgrades are not presently supported. Previous version of the BPF Web Application need to be un-deployed first using the Web Application Server administrative console, before installing a new version.

Install Shield based deployment

BPF COM applications are installed and uninstalled running the `filenet_bpf_explorer_setup.exe` Install Shield setup.

Versioning

Generating version numbers

All BPF components are versioned when compiled using the format that matches the FileNet P8 Platform, (such as BPF 3.5.2).

Retrieving version information

The version number is stored in the JAR manifest file in each BPF Jar file (`Manifest.mf`).

The BPF Web Application version number is retrieved via the following URL:

`http://server:port/bpf/About.jsp`

Error and trace logging

Error logging refers to logging of server activities in such way that a server administrator or an operator can understand and perform self-help when problems arise. Trace logging refers to verbose logging of server activities in such a way that only support personnel or programmers need to understand and perform debugging based on the given log.

Error logging via Log4j

With log4j, it is possible to enable logging at runtime without modifying the application binary. Editing a configuration file, without touching the application binary, can control logging behavior.

BPF Business Services utilizes Log4j to implement error logging. The logging level of the BPF Web Application is set to Warning and could be changed at runtime. Setting the logging level to debug will enable a detailed output.

Logging in the BPF Web Application is turned on by default with the Warn level.

These are the most commonly used values:

- error - log errors only
- warn - log errors and warnings (default)
- Info - log all above and information type messages, which includes business level data (Case ID being created, etc)

debug - log information on the detailed level (this option may generate large file)

The full list of the logging levels follows:

all, debug, info, warn, error, fatal, off

By default, logging for the Web Application is directed to the `bp8.log` file in the `WEB-INF/logs` folder in the deployed Web Application. The log file will be renamed to `bp8.log.YYYY-MM-DD` and the new empty log created at the end of each day.

To modify the logging level or specify a different log file name:

Edit `log4j.xml` file.

To change the Web Application log file name, modify the following section:

```
<param name="File" value=" ${bp8.app_root}/WEB-INF/logs/bp8.log"/>
```

To change the logging level of any of the log files, modify the setting in the corresponding group:

```
<level value="info"/>
```

To turn profiling off, set the following value:

```
<level value="off"/>
```

NOTE The Web Application does not need to be restarted in order for changes in the `log4j.xml` file to take effect.

Logging Configuration Options

OPTION #1 - File Size

You can set rollover to occur based on the file size.

RollingFileAppender extends FileAppender by limiting the size of log files to some user specified length. Logging output is written to the file name specified by the File option. When the log file reaches the specified size, it is rolled over. It is renamed by appending 1 to the file name. If a .1 file exists, it is first renamed to .2 and so on.

For example, if the File option is set to `wombat.log`, then `wombat.log` will be renamed as `wombat.log.1`. Any existing `wombat.log.1` file is renamed as `wombat.log.2`, any previously existing `wombat.log.2` file is renamed to `wombat.log.3` and so on, until `MaxBackupIndex`.

For instance, assuming `MaxBackupIndex` is set to 4, `wombat.log.4` is simply deleted without further cascading.

Thus, in addition to the FileAppender options, RollingFileAppender has two additional `MaxFileSize` and `MaxBackupIndex`, as summarized below.

The `MaxFileSize` option takes a string value representing a long integer in the range 0 - 263. You can specify the value with the suffixes KB, MB, or GB so that the integer is interpreted as being expressed respectively in kilobytes, megabytes, or gigabytes. For example, the value 10KB will be interpreted as 10240.

Rollover occurs when the log file reaches `MaxFile Size`. Note that since the last log event is written entirely before a roll over is triggered, actual files are usually a bit larger than the value of `MaxFileSize`. The default value of this option is 10MB.

The appender will look like:

```
<appender name="R" class="org.apache.log4j.RollingFileAppender">
  <param name="File" value=" ${bp8.app_root}/WEB-INF/logs/bp8.log"/>
  <param name="MaxFileSize" value="1MB"/>
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d %-5p %C{2} - %m\n"/>
  </layout>
</appender>
```

OPTION #2 - Time Based (Default)

You can set rollover to occur based on the file size.

DailyRollingFileAppender extends FileAppender so that the underlying file is rolled over at a user chosen frequency. The rolling schedule is specified by the `DatePattern` option. This pattern should follow the `SimpleDateFormat` conventions. In particular, you must escape literal text within a pair of single quotes. A formatted version of the date pattern is used as the suffix for the rolled file name.

For example, if the File option is set to `/foo/bar.log` and the `DatePattern` set to `'yyyy-MM-dd'`, on 2001-02-16 at midnight, the logging file `/foo/bar.log` will be copied to `/foo/bar.log.2001-02-16` and logging for 2001-02-17 will continue in `/foo/bar.log` until it rolls over the next day.

It is possible to specify monthly, weekly, half-daily, daily, hourly, or minutely rollover schedules.

<u>datePattern</u>	<u>Rollover schedule</u>
'.'yyyy-MM	Rollover at the beginning of each month At midnight of May 31st, 2002 <code>/foo/bar.log</code> will be copied to <code>/foo/bar.log.2002-05</code> . Logging for the month of June will be output to <code>/foo/bar.log</code> until it is also rolled over the next month.
'.'yyyy-ww	Rollover at the first day of each week. The first day of the week depends on the locale. Assuming the first day of the week is Sunday, on Saturday midnight, June 9th 2002, the file <code>/foo/bar.log</code> will be copied to <code>foo/bar.log.2002-23</code> . Logging for the 24th week of 2002 will be output to <code>/foo/bar.log</code> until it is rolled over the next week.
'.'yyyy-MM-dd	Rollover at midnight each day. At midnight, on March 8th, 2002, <code>/foo/bar.log</code> will be copied to <code>/foo/bar.log.2002-03-08</code> . Logging for the 9th day of March will be output to <code>/foo/bar.log</code> until it is rolled over the next day.
'.'yyyy-MM-dd-a	Rollover at midnight and midday of each day. At noon, on March 9th, 2002, <code>/foo/bar.log</code> will be copied to <code>/foo/bar.log.2002-03-09-AM</code> . Logging for the afternoon of the 9th will be output to <code>/foo/bar.log</code> until it is rolled over at midnight.
'.'yyyy-MM-dd-HH	Rollover at the top of every hour. At approximately 11:00.000 o'clock on March 9th, 2002, <code>/foo/bar.log</code> will be copied to <code>/foo/bar.log.2002-03-09-10</code> . Logging for the 11th hour of the 9th of March will be output to <code>/foo/bar.log</code> until it is rolled over at the beginning of the next hour.
'.'yyyy-MM-dd-HH-mm	Rollover at the beginning of every minute. At approximately 11:23.000, on March 9th, 2001, <code>/foo/bar.log</code> will be copied to <code>/foo/bar.log.2001-03-09-10-22</code> . Logging for the minute of 11:23 (9th of March) will be output to <code>/foo/bar.log</code> until it is rolled over the next minute.

Do not use the colon (:) character anywhere in the DatePattern option. The text before the colon is interpreted as the protocol specification of a URL, which is probably not what you want.

NOTE For more details on log4j logging, refer to <http://logging.apache.org/log4j/docs/>.

Trace logging via AspectJ

AspectJ is an extension to the Java programming language that adds to the Java aspect-oriented programming (AOP) capabilities. AOP allows developers to gain the benefits of modularity for concerns that cut across the natural units of modularity.

BPF Business Services utilizes AspectJ to implement trace logging. All functions within the core BPF Business Services Java classes can be traced to capture the execution times.

Action dispatcher logging

The BPF Web Application Action Dispatcher trace logging allows capturing all XML message traffic between the web browser client and the BPF Web Application. Incoming and outgoing XML messages are captured in the `\logs` folder in the deployed web application. Logging can be enabled for specific users only.

User activity logging assists with application performance evaluation and troubleshooting various issues by capturing XML message traffic between the client and the server.

User activity logging is enabled via the following setting in BPF Explorer:

Section name: XML Logging

Setting name: Enable logging XML actions

Value: True

Specify user logon names for the users you want to log the activity for via the following setting in BPF Explorer:

Section name: XML Logging

Setting name: User logon names to log activity

Value: <user names, comma separated>

Example: ceadmin, peadmin

If you have Metadata Cache turned on, you need to stop and start the Web Application for these settings to take effect.

A new file will be created for each request or response message and be captured in `WEB-INF/logs` folder in the following format:

`$userName_$actionName_$direction_$uniqueNumber.xml`

where:

`$userName`: User logon name

`$actionName`: Web Application function name

`$direction`: Response or Request

`$uniqueNumber`: Current time in milliseconds

Utilities

Workflow Region Maintenance tool

The BPF Workflow Region Maintenance tool (wfutil) provides a method to import workflow region configuration from an XML file. This tool allows the following:

- Creates public queues
- Exposes system and user defined properties on the queues, rosters, and event logs
- Creates indices on the queues, rosters, and event logs

Once deployed, the wfutil folder will contain the following content:

- `wfutil.jar` – wfutil main application
- `wfutil.cmd` – Windows version of the script executing wfutil
- `wfutil.sh` – UNIX version of the script executing wfutil
- `doc` (folder) – JavaDoc reference and a sample configuration file (`qcreate.xml`).
- `logs` (folder) – Folder for the output log files
- `lib` (folder) – Folder for BPM API dependency files (`eProcess.jar`, `mailapi.jar`, `xerces.jar`)

Workflow Map Import tool

The BPF Workflow Map Import tool allows importing the workflow process map queue, response, and step names to the BPF Metastore from the PEP file.

Notices

This information was developed for products and services offered in the U.S.A.

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

IBM 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:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION 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 states 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. IBM 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-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply 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:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

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 IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM'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 the names and addresses used by an actual business enterprise 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 IBM, 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. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

IBM

DB2

System Storage

FileNet is a registered trademark of FileNet Corporation, in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation 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 company, product, and service names may be trademarks or service marks of others.



Program Number: 5724-R75

Printed in USA

GC31-5520-00

