# Merlin Feature List

## Abstract

This is the first public review draft of the J2SE Merlin Release feature list.

This specification will define the target feature and API set for Merlin. The goal is that the final Merlin release will include all the high and medium priority features from this target list, but depending on implementation and API design schedules some items may be deferred to a later release if they are unable to make the release schedule for Merlin. The final specification will reflect the final Merlin deliverables.

Please try and send first round comments by July 20th to merlin-jsr@eng.sun.com

*Version 0.30*

*Sun Microsystems*
*June 14, 2000*

# Contents

# 1 Introduction

This specification will describe the feature and API set for the next update release of Java 2 Standard Edition, code named “Merlin”, targeted to ship in 2001. Merlin is one of a series of update releases to J2SE.

The intention is to ship update releases on a regular 12-18 month cycle, with each update release including a combination of quality improvements and a small amount of new features. In assessing features for Merlin it is necessary to balance the schedule goals for Merlin against the desirability of individual features.

## 1.1 Themes

It is expected that most of the effort in the Merlin release will be around product quality (bug fixing) and product performance with only a relatively small amount of new features and new APIs.

The over-arching goal of Merlin is to support the deployment of enterprise class Java applications. The release is therefore targeted at the following major themes:

- Reliability
- Serviceability
- Scalability
- Performance
- Deployment

Merlin will be fully compatible with previous J2SE releases.

## 1.2 Granularity

This specification is only intended to cover “substantial” features that have significant API impact or which involve significant engineering or test effort. It is not intended to cover individual bug fixes or minor RFEs.

As part of the initial work on Merlin the J2SE engineering teams are reviewing all open bugs and RFEs, with the goal of addressing as many as possible of these in Merlin.

This specification will not itself define any new APIs, rather it will reference APIs defined in other JCP Specifications or through the JCP maintenance process.

## 1.3 Prioritizing Features

Target features for Merlin are prioritized into three buckets: Release Drivers, Group Drivers, and Targets of Opportunity.

### 1.3.1 Release Drivers

Release Drivers are those features that are considered essential to the success of the release. They will be given the highest priority during the development phase, and if necessary the release may be delayed in order to accommodate delays in developing a Release Driver.

Because Release Drivers can potentially delay the release, we would like to be very selective in designating features as Release Drivers.

### 1.3.2 Group Drivers

Group Drivers are features that will be treated as priority items within engineering teams. These features will take second place to Release Drivers, but will be tracked carefully at both the individual team level and the overall release management level.

The intention is that all Group Drivers will be included in the final release, but the release will not be delayed in order to accommodate a delayed Group Driver. Thus some Group Drivers may be dropped from the release.

We need to be careful to manage the number of Group Drivers so that engineering teams have time to address their open bug lists and their performance items as well as all their Group Drivers.

### 1.3.3 Targets of Opportunity (also known as "General Features")

Targets of Opportunity are lower priority features than Release Drivers and Group Drivers.

The intention is that after engineering teams have resolved their bug backlogs and have implemented their Release Drivers and Group Drivers they will move on to implementing Targets of Opportunity.

It is expected that in practice a large percentage of the Targets of Opportunity will be accommodated in the release, but this will vary from area to area, depending on the teams' progress in addressing their other work. The release will not be delayed to accommodate Targets of Opportunity.

## 1.4 Note to Reviewers

We are interested in your input on whether other features should be included in the release, whether features need to be re-prioritized, and whether features need to be dropped.

A major challenge for us in defining an update release like Merlin is managing the feature set so as to meet the main needs of the broad Java developer community without adding so many features that the release becomes unmanageable. After previous experiences with overly large releases (notably J2SE 1.2), the J2SE planning team is working hard to ensure that future update releases are of a more manageable scope, so that they can be delivered on schedule with high quality.

Therefore in requesting new features or in requesting that feature priorities be raised, please provide us with:

- Information about why the feature is broadly useful.
- Information on what kinds of new or different applications the feature might enable.
- Why does this need to be in J2SE rather than in a third party library?
- What could we drop from the release to accommodate the new feature?

Thanks,

The J2SE Planning Team.

# 2 XML

### XML Parser and DOM 1.0

**ID: 4290542 Release Driver**

JSR 005 has defined an XML standard extension for the Java platform based on the W3C DOM 1.0 API and the SAX parser 1.0 API.

An implementation of these important APIs should be bundled in the JRE.

See also Merlin feature 4324873 "DOM 2.0 and SAX 2.0".

---

### XML Data Binding

**ID: 4286947 Group Driver**

http://java.sun.com/aboutJava/communityprocess/jsr/jsr_031_xmld.html

The proposed specification will define an XML data-binding facility for the JavaTM Platform. Such a facility compiles an XML schema into one or more Java classes. These automatically-generated classes handle the translation between XML documents that follow the schema and interrelated instances of the derived classes. They also ensure that the constraints expressed in the schema are maintained as instances of the classes are manipulated.

The proposed specification will vastly simplify the creation and maintenance of XML-enabled Java programs. Data binding automatically maps the components of an XML document to in-memory objects that represent, in an abvious and useful way, the document's intended meaning according to its schema. This allows Java programs that manipulate XML content to be written at the same conceptual level as the content itself, rather than at the level of parser events or parse trees.

The only existing Java APIs for manipulating XML are the low-level SAX parser API and the somewhat higher-level DOM parse-tree API. WHile it is possible to write XML-enabled programs using these interfaces, doing so is likely to be tedious and error-prone. The resulting code is also likely to contain many redundancies that will make it difficult to maintain as bugs are fixed and as the relevant schemas evolve.

---

### DOM 2.0 and SAX 2.0

**ID: 4324873 Group Driver**

Feature 4290542 describes the 1.0 versions of the SAX parsers and of the W3C DOM APIs.

2.0 versions of these APIs are also under development. If these APIs are final in time for Merlin beta, then we should provide implementations in Merlin.

Details will be resolved through a JCP JSR.

---

### Javadoc XML doclet

### ID: 4292100 Target of Opportunity

This feature encapsulates a doclet that produces XML output instead of html.

This will provide a standard, parseable interchange format for static program information. It allows design and modeling tools to easily grab and display stored information, while facilitating efficient translation of API docs using common XML tools. It should allow full and incremental builds of Javadoc from XML rather than .java source files

---

### XSLT support

### ID: 4324875 Target of Opportunity

XSLT is a standard for transforming XML.

An XSLT implementation should be delivered in the JDK. If possible this should be based on work under way in the open source community.

The APIs will need to be defined through a JCP JSR.

---

# 3 AWT

---

### support "headless" Java

**ID: 4281163 Release Driver**

'Headless' operation refers to the ability to run a Java program without a monitor, mouse or keyboard. In unix environments, it usually means that X or an equivalent windowing system does not have to be present.

Headless Java operation would mean that a server side Java program should be able to use the full J2SE or J2EE API without a GUI environment being present. This means that all AWT calls to manipulate components, graphics, fonts, printing etc. should be functional.

---

### File Dialog Extensions for Swing, Windows LAF

**ID: 4290709 Release Driver**

Although Swing's JFileChooser closely resembles the Windows common file dialog under the Windows look and feel, there are several deficiencies that customers complain about. In particular, the Microsoft Windows aesthetics guide mentions the following:

> "If you cannot use the [common] Open and Save As dialog boxes, you should incorporate the following features into your open and save dialog boxes to ensure that they are consistent with the shell, the Windows accessories, and other applications:
>
> + Support the same namespace hierarchy as the shell; that is, Desktop should be at the root of the hierarchy, followed by all folders and objects on the desktop, including My Computer, My Network, and so on. [...]
>
> + Support shortcuts (also known as shell links). [...]
>
> + Display filenames with the corresponding icons and filename extensions removed, as the shell does.
>
> + Allow the user to browse the network hierarchy directly.
>
> + Make sure that all of your dialog boxes (not just your open and save dialog boxes) use only nonbold fonts. [...]"

The first four points are not possible in Swing without additional API support from the AWT. The AWT common file dialog is also currently an unacceptable solution for the following reasons:

+ It cannot be embedded into a frame or dialog like JFileChooser.

+ Filename filter is not implemented and is difficult to implement on Windows. (see file dialog filtering proposal).

---

### drag and drop enhancements

**ID: 4290723 Release Driver**

Minimal support for AWT Drag and Drop was added in JDK 1.2 and was further enhanced in 1.2.2. This provides a gesture based solution, with clear feedback mechanisms, for copying or moving data (text, graphics, files...) from one application or directory etc. to another.

In Merlin, Drag and Drop should be further developed and should provide consistent deadlock-free, support for drag and drop:

between a VM and the native platform
within a VM
between VMs

All supported MIME types should be draggable and the application should be able to determine the best way to utilize the copied or moved information.

---

### Native drawing JAWT interface improvements

**ID: 4281429 Group Driver**

The new native drawing support in the Kestrel (1.3) release is a welcome addition and will likely see use in the market.

This feature works as advertised, however there is an area of difficulty that the programmer has in coding the native side of the component. The problem has to do with management of allocation of various native resources. The AWT implementation already has methods in its native implementation to manage these resources, but access to those methods is not available to one programming to the JAWT interface. This means, then, that the programmer must separately manage allocation of those same resources.

The project proposal is to extend the native JAWT structures to have additional methods. The extension must be on a per platform basis since both the specific resources to coordinate, and the data types to use, are unique to each platform.

---

### Clipboard Fixes and Enhancements

**ID: 4289847 Group Driver**

The main problem with the current clipboard is that it does not allow non-String data to be transferred between VM's, even though the API allows arbirtrary objects to be put on the clipboard. This needs to be fixed ASAP (maybe even in Kestrel?) to serialize/deserialize objects for inter VM transfer.

More on the enhancement side, the clipboard does not provide conversions for common data types.
For example, you cannot transfer images between Java and native applications. We should provide two way conversions for common data types e.g.:

- Java Image objects <-> native bitmap formats (BMP, PICT, X pixmap)

- Styled text <-> RTF or whatever
- Sound clips <-> native audio format
- etc etc.

We should also allow Java applications to access unknown data types as a raw byte stream in case they want to provide their own conversions. Whatever data exchange formats we provide for the clipboard should also be supported for drag and drop.

---

### Focus Management Enhancements

### ID: 4290675 Group Driver

The current AWT focus subsystem in woefully inadequate. It suffers from major design and API problems, as well as over a hundred open bugs. Many of these bugs are caused by platform inconsistencies, or incompatibilities between the native focus system for heavyweights and the Java focus system for lightweights. Developers do not have access to a comprehensive focus specification, nor does the AWT team have any substantive documentation of the implementation. We propose to address these problems in the following way :

Documenting the existing focus implementation, both from an internal and external view point.
We will compose a detailed, formalized focus specification, which addresses the shortcomings of the existing focus APIs.
Development of the specification will be accompanied by construction of native, proof-of-design prototypes.
We will proceed to implement this new specification as completely as possible.

---

### No way to create a frame without decorations.

### ID: 4038769 Target of Opportunity

This is really on all platforms. There is currently no way to create a main/frame window without decorations. There are a number of solutions to this, the easiest being to provide a manifest constant or method in Runtime the allows retrieval of the Desktop window. That could then be passed as the parent to the Window() constructor. Other ways are more complicated ...

---

### Mouse Wheel Support

### ID: 4289845 Target of Opportunity

Wheel mice, with a scroll wheel as a middle mouse button, are increasingly popular. Anecdotal evidence from JavaOne indicates that developers would like Java to provide support for such devices ASAP.

A new type of mouse event/listener could be introduced to track mouse wheel movement. Mouse wheels are supported on Win 2k and on older Windows versions via the add-on Intellimouse software. X has various ways of supporting wheel mice as well:
http://www.inria.fr/koala/colas/mouse-wheel-scroll/

---

### getListener Project Completion

### ID: 4290704 Target of Opportunity

The implementation for getListener() was added as a new feature in the Kestrel release by the Swing team. The implementation of getListeners() for JComponent neds to be completed for the following listeners (see bug 4257538):

PropertyChangeListener, AncestorListener, VetoableChangeListener

The fix will be implemented in AWT for each of the following components:

java.awt.Component
java.awt.Container
java.awt.Scrollbar
java.awt.Checkbox
java.awt.Window
java.awt.List
java.awt.TextComponent
java.awt.Choice
java.awt.TextField
java.awt.Button

And also for the Swing classes:

javax.swing.JComponent
javax.swing.tree.DefaultTreeSelectionModel

---

### SystemFlavorMap Enhancements

### ID: 4304018 Target of Opportunity

Don't force SystemFlavorMap to build its map from a properties file. On some systems, the mapping of data flavors to MIME types can be changed dynamically -- not just by modifying the mappings, but more frequently by adding and removing items. Since SystemFlavorMap currently only supports using the contents of flavormap.properties, it doesn't reflect any of the changes that a user may make to a centralized OS-level mapping.

---

### Allow access to AWT Peers' "private" variables and APIs

### ID: 4304020 Target of Opportunity

Allow Java "public" access to the currently "private" Peer variables and APIs that are used extensively by the C versions of Peers. Our Peers are pure Java, we have to obey the restrictions.

---

# 4 Java 2D

---

### New pipeline architecture needed to reduce overhead of common operations

**ID: 4228939 Release Driver**

Many common operations performed on a Graphics object will interrupt the rendering process by causing rendering pipeline invalidation and the de-caching of important rendering information. While such validation may be necessary in some situations, the validation steps are executed very conservatively in the 1.2 implementations.

For example, the rendering data will be invalidated when the color is changed from one opaque color to another, when a graphics object is cloned using the create() method, when a simple integer coordinate translation is requested, and many other operations which do not fundamentally change the nature of the rendering algorithms needed to respond to the various rendering requests.

Since Swing hierarchies typically rely heavily on using the create(), setColor(), translate(), and clip() methods this can waste a lot of time during application repaints while the many graphics objects have their rendering data re-created over and over again during pipeline invalidations and subsequent validations.

---

### Surface based on the new pipeline architecture needed to speed up Remote X

**ID: 4317185 Release Driver**

We need to create a new type of SurfaceData based on the new pipeline architecture that was put in as part of RFE 4228939, to speed up rendering on remote X11 displays. This is a shadow feature of bug 4204845 : Remote use of double buffering on JDK1.2 is very slow.

---

### Add a pluggable Image I/O framework into Java 2 SE SDK

**ID: 4101949 Group Driver**

This RFE is being used to represent the feature work necessary for a pluggable Image I/O framework in the Java 2 Standard Edition SDK

This RFE subsumes the following requests:-

4029964 RFE: support loading of TIFF images additionally to GIF & JPEG
4055931 Provide more image file format capability natively
4075967 Add-on Image Formats
4095978 Provide API to retrieve individual frames of a multi-frame GIF
4098085 Import/Export filters for graphics
4127913 Create a GIF file from a Image object
4144824 Support for JPEG 12 bit image decompression
4191048 RFE: gif image encoder classes
4216579 Can we have methods to start/stop animated gifs
4285169 Add Image I/O framework

See the Java Specification Request for more details

http://java.sun.com/aboutJava/communityprocess/jsr/jsr_015_iio.html

---

### Add hinting to TrueType scaler

### ID: 4227239 Group Driver

Add the ability to produced hinted glyphs from TrueType fonts.

---

### Additional Java 2D printing work

### ID: 4285177 Group Driver

We need to fix printing so that it works in all cases. This includes printing components, printing in plug-in, and more GDI/Postscript support. We need to try to unify the code for 1.1 and 1.2 printing wherever possible because there is a lot of redundant work and we end up solving the same problems.

We also need to track what is going on in JSR 6, Unified Printing API. This may entail adding new classes to better support the JINI printing model (e.g. adding an Attributes class, similar to Job/Page Attributes in java.awt).

==========

The Java 2D printing API was introduced in Java 2 SE, Version 1.2. The API was designed to support sophisticated document printing, but time constraints meant that the implementation was primarily focused on being able to print all 2D graphics.
In subsequent releases (V 1.2.2 and V1.3) there were implementation enhancements only, focused on improving printing performance for many typical applications, such as web pages, reports etc, which involve text, some small number of graphic images, and occasional simple 2D graphics.

The result of this work is that 2D printing in 1.3 is hugely improved in robustness, completeness, quality and performance compared to earlier releases.

Moving forward in releases after Java 2 SE, Version 1.3, the 2D printing API needs to be enhanced further in implementation, and also to include new APIs to support more sophisticated printing applications.
The need for these new APIs is demonstrated by numerous RFE's and also the formation of the Unified Printing Expert Group (JSR-006) and the associated work of the JINI printing working group.

The following are the currently expected items to be implemented for the "merlin" release of Java 2 SE.

1) 2D printing will be enhanced in a number of ways
   a) additional page control and attributes
   b) additional job control and attributes
   c) printer enumeration & description
   d) printer capabilities queriying (such as duplex capability)
   e) cross platform print and page dialogs
   f) improved quality and consistency in page imaging
   g) support component style printing (AWT printing)

2) Enhancements 1(a) to 1(d) above to the 2D printing API will be designed to

integrate well with the results of the JINI printer working group and JSR-006 expert group.

3) As a benefit of 1(g) above, AWT's page imaging and job control will be reimplemented on top of 2D printing to reduce implementation size & duplication. This will probably be via means of an adapter layer which could also be the foundation for more easily integrating alternate implementations for accessing JINI enabled printers, or allowing a printer driver API implementation on Solaris which might generate output suitable for other printer languages than PostScript. Merlin is not expected to deliver such capabilities, only to lay the foundation for this in a later release.

Apart from the work directly on 2D printing, we also expect other groups to begin to make more direct use of the 2D printing API in merlin as follows:

1) Appletviewer will use 2D printing API so it can print applets which use 2D graphics.

2) Selected Swing components will be enhanced to directly support printing via the 2D printing. ie a printer friendly view of the model.

3) Headless Java will support 2D printing in an environement with no X-display

These latter features need commitment from groups other than 2D, but we believe such commitments have been made.

---

## Add access to bidirectional text information

## ID: 4285083 Target of Opportunity

The Unicode Bidirectional Algorithm is is currently implemented in two places, sun.awt.font and in javax.swing.text. There should be one public implementation in java.text. Swing wants to use a public API to get this information, and the duplication of code is undesirable.

The Unicode Bidirectional Algorithm analyzes text using the Unicode character properties, and determines the direction of runs of the text. This is necessary to properly display Hebrew and Arabic text in the correct order.

The current implementation is all Java, but it would be useful to have efficient access from the native font code so that Hebrew and Arabic text can be more efficiently rendered. We would use a native implementation and provide Java access through JNI. We have a C++ implementation available.

The java.lang.Character class does not provide access to a character's bidirectional property. As Character is the interface for all Unicode character information, it should be extended to add access to this information for Java clients. This is requested in bug 4132045.

---

### Upgrade Lucida fonts with hints

### ID: 4285089 Target of Opportunity

The Lucida fonts that are in the Java 2 SDK should be upgraded to contain hints. This would give the Java 2 SDK higher quality fonts that could be used in place of existing fonts or if no other fonts are available.

---

### Improved Text Layout Support

### ID: 4285090 Target of Opportunity

Text layout encompasses a number of services required for properly rendering and manipulating text in various languages. Such services include choosing the appropriate positions and shapes of character glyphs, measuring dimensions of text, hit testing, cursor movement, and so on. Many languages require features not commonly required for simple English rendering, such as reordering, ligatures, accent placement, and multiple character shapes.

Text layout depends on information from the font system, principally java.awt.Font and java.awt.font.GlyphVector. Layout is performed by the graphics system (java.awt.Graphics and java.awt.Graphics2D) to render text, and by java.awt.font.TextLayout to both render text and provide metric, hit testing, and cursor movement data. It is also performed by Swing text components (javax.swing.text.JTextComponent).

There are several areas that need improvement:
- accents are not properly positioned
- languages requiring ligatures or contextual reordering aren't supported
- language-appropriate rendering is not performed as the language and locale are not identified as part of the rendering process
- Swing's only public access to layout support is through TextLayout, which causes inefficiencies when both perform the same text analysis
- there are multiple internal text pathways, leading to maintenance difficulties, inefficiences, unwanted variations in behavior, and bugs
- composite fonts introduce problems in layout and rendering at the native level

The proposal is to stage a solution to these problems over time. The initial steps are to enhance the internal native implementation to enable more use of font information in performing layout, and to provide public API to more layout processes and data so that Swing can access it. Later steps involve resolving the multiple internal text paths, explicitly distinguishing and formalizing the distinction between composite (font.properties) fonts and 'real' fonts, and more clearly distinguishing in the native code between the functions of fonts, font strikes, glyph caches, rasterization, and layout.

---

### Additional cleanup of font-related code

### ID: 4285092 Target of Opportunity

```
There are a number of areas in the font code that would
benefit from restructuring/cleanup.  These include:
- font files
- font cache
- glyph vector
- strikes
- font metrics
- composite font


==========================================================================
```

Handling of font files opening/closing needs to be more efficient. Certain font data needs to be cached. Files not used need to clean up and be closed.

The font code dealing with composite fonts needs to be cleaned up - this would involve simplification to the strike handling and simplify the complexity.

Font substitution and default fallback mechanisms for fonts and characters substitution needs to be improved. Style handling for both algorithmic and regular style substitution also needs to be improved.

Some of the glyphvector and metrics cleanup will also give more accurate values for stuff like metrics, baselines and transformations.
Some of this work is in progress.

---

### OpenType font table support

### ID: 4285161 Target of Opportunity

```
We need to define an architecture for providing general
OpenType font support.  This provides international
character support for contextual scripts like Thai,
Indic, Arabic, Hebrew, etc. It also provides enhanced typographical
support for Roman languages.
```

---

# 5 Swing

### Drag and Drop Support for Swing Components

**ID: 4290983 Release Driver**

While support for Drag and Drop has been available since JDK 1.2, usage in Java Foundation Classes required unique developer implementations on a per component basis. In JDK1.3 default Drag and Drop implementations will be made available on JText, JList, JTree, JLabel, and JTable components. Additional APIs will be added to activate alternative Drag and Drop implementation or to disable DnD on a component.

---

### JFileChooser Update

**ID: 4292456 Release Driver**

The Windows look and feel implementation will be brought more closely in line with the native Windows file chooser. This work is largely covered by feature 4290973. This feature depends heavily on support from the AWT, see feature 4290709.

Presently, JFileChooser allows one to specify a FileFilter object that can be used to limit the files the file chooser displays. A default implementation of the abstract FileFilter class was not provided because there wasn't a file system independent way to compute the "type" of a file. Feature 4313887 - "New I/O: Improved filesystem interface", adds support for discovering a files MIME content type. We'll add support for filtering files based on this feature, probably by providing a new FileFilter subclass.

---

### improve Swing peformance

**ID: 4294618 Release Driver**

J2SE 1.2 introduced a serious performance degradation when displaying Swing-based applications either on local or remote displays, i.e. JDK 1.1 + Swing 1.1.1 was much faster than 1.2.

J2SE 1.2.2 and 1.3 substantially improved Swing performance, and 1.3 is now typically faster than JDK 1.1.8 for local Swing applications. However there are still performance issues for remote displays.

We should continue to address Swing performance for both local and remote display. Merlin 2D feature 4317185 will address remote display issues. Merlin feature 4228939 will boost underlying 2D performance for common swing operations. We should also continue with general performance work in Swing itself.

---

### Improved scrolling

**ID: 4285182 Group Driver**

Swing has made major improvements in scrolling for Kestrel. However, the improved scrolling algorithms fail under some conditions. This causes the viewport to be forced back to a very conservative slow mechanism.

The problems lie in the AWT Graphics.copyArea() method, and how it produces update events asynchronously. If a new version of this method were provided then the Swing viewport could be modified to be even more efficient.

---

### New Swing Component: Spinner

**ID: 4290529 Group Driver**

The most commonly requested Swing component is a spinner - a single line input field that lets the user select an integer or an object value from an ordered set. Spinners typically provide a pair of tiny arrow buttons for stepping through the possible values, the up/down arrow keys also cycle through the values. The user may also be allowed to type a (legal) value directly into the spinner. Although combo boxes provide similar functionality, spinners are sometimes preferred because they don't require a drop down list that can obscure important data.

---

### JProgressBar - support for "indeterminate" display

**ID: 4290717 Group Driver**

This functionality can be implemented by adding API to the existing JProgressBar implementation. When the programmer specifies that the task has an indeterminate duration then the progress bar, instead of displaying its usual left-to-right or bottom-to-top animation, would display continuous animation such as a swirling barber pole. The intent would be to show the user that the program is doing something, even though the actual progress of the task can't be determined until it either finishes or fails.

---

### Windows 2000 L&F

**ID: 4290973 Group Driver**

The goal of this work is to provide an updated Windows Look&Feel implementation which emulates (to the best of our ability) the Windows Look&Feel provided by Microsoft in Windows95, Windows98, and Windows2000. This work includes adding support for new features introduced in Windows98 and Windows2000, as well as improving our general Look&Feel emulation for all versions of the Windows Look&Feel.

This project will incorporate support for more complete tracking of the Windows desktop properties set by the user (colors, font, sounds, etc)

including tracking dynamic changes at runtime.

This work will be done in conjunction with the HI group to measure our compatibility with Windows 2000 and to provide the updated JLF.

---

### JTabbedPane Update

**ID: 4303623 Group Driver**

When all of the tabs added to a JTabbedPane do not fit on a single row, extra rows of tabs are created. Many users and developers have observed that one row of tabs is easy to use but two or more can be confusing. This may be because the physical analogy to a stack of folders starts to unravel when there's more than one row of tabs, or simply because finding a single tab label in a grid is more taxing than finding a label on single line. Providing an arrangement of tabs that's easier to use when all of the tabs don't fit on a single row is important because many applications create tabs dynamically and can't easily ensure that they'll all fit on a single row. We plan to extend JTabbedPane to support another layout option for this case, for example a single scrollable row of tabs.

---

### Global Menu Bar, Macintosh Look and Feel

**ID: 4303635 Group Driver**

The Macintosh merges the selected windows menubar with the system menubar that's always visible at the top of the screen. The Swing JRootPane container, which is the default (only) child of JFrame, JDialog, and JWindow, always locates the menubar in a horizontal band above its contentPane child. This matches the layout of menubars for most look and feels, but not the Macintosh. We propose to modify Swing so that the MacLookAndFeel from Apple can merge Swing menubars with the Apple desktop menubar.

---

### Optimize Memory Usage in Swing HTML classes

**ID: 4286884 Target of Opportunity**

The Swing HTML class need to be optimized to use less RAM. Currently the JEditorPane uses a very large amount of memory when medium to large documents are loaded into the pane. There are numerous ways this could be improved. This may also have a positive effect on footprint for any styled text, not just HTML.

---

### Metal2D (aka Java look and feel 2.0): Updated version of default J2 SDK SE L&F

**ID: 4290471 Target of Opportunity**

What is the feature and what problem is it solving?
Offering an updated Java look and feel that equals or surpasses other GUI toolkits in terms of ease of use, contemporary aesthetics and compatibility

with the changing graphics and display capabilities of the standard hardware platforms.

How will it work?
We will subclass the current Java Look and Feel utilizing the advanced graphics capabilitis offered by the Java2D API where appropriate.

What are the benefits over other solutions?
There are no other solutions offered for this feature.

---

## Auditory Feedback for Swing Components

### ID: 4290988 Target of Opportunity

Currently, the Swing components do not provide the same auditory feedback as the native components on many platforms. We plan to extend the basic Swing PLAF module, so that derived look and feel classes can provide audio feedback by configuring Swing defaults table entries. This work, along the "Windows L&F Update" project (4290973) will help ensure that Swing applications integrate seamlessly with native applications.

---

# 6 Accessibility

### Add mnemonic support for accessing tabs on a JTabbedPane

**ID: 4284674 Target of Opportunity**

The feature makes it possible for developers to add mnemonics to the tabs of pages contained in a JTabbedPane. This is a feature request of many in the development community, it's corresponding bugid is 4200562.

It allows the user to navigate directly to a tabbed page simply by pressing Alt+the mnemonic character simultaneously.

It's more direct than the current solutions which follow. Activating a mnemonic is well understood by users and the user need only "look" at the UI to know what to do as opposed to the following methods which require remembering non-obvious key sequences or greatly reduced navigation efficiency.
First way - the current fastest way requires a Control+up arrow then arrow key navigation to the desired tab.

Second way - If the user forgets this non-obvious method they must tab though all active components contained in the tabbed page until focus is given to the tab then arrow keys are used to move to the desired tab.

---

### Add support for first letter component navigation in list like components

**ID: 4284709 Target of Opportunity**

JMenu*, JComboBox, JPopupMenu, JHtmlPane (the links), JTree, JList, and JDesktopPane are components that present the user with a "list" of objects to choose from. These objects could be the names of features in a menu or folder icons in a file directory. The problem it solves is it allows the user to efficiently traverse long lists of data to get to a particular entry or object (e.g. printer Teabag from a list of all printers in CUP02).

What this feature does is move the keyboard focus to the first object whose first letter matches the alphanumeric key pressed by the user. Subsequent same key presses move the keyboard focus to the next object that starts with the same letter.

Other keyboard based features require the user to navigate thru each of the possible choices presented to the user to get to the one they want. This can be very tedious if the list of choices is very long.

---

### Implement AccessibleDocument on the text package

### ID: 4284736 Target of Opportunity

AccessibleDocument is an interface to Java applications to enable an Assistive Technology Vendor application to control the look of a document to improve readability as well as edit a document from alternative input programs. It basically takes subset of functionality already in text packages and codifies it into a standard, clearly defined interface for assistive technologies to easily find and interact with.

The interface will mostly be methods that call already existing methods in text package which assistive technologies will call thru the Java Accessibility Utilities Sun provides.

The benefit is assistive technology vendors of tools won't need to learn the text component or have to dig thru the various components that make up a text package to find the methods their soution needs.

---

### For Accessibility: Allow Swing components to work better with IDE's

### ID: 4285960 Target of Opportunity

This feature is make it easy for IDE's to enhance their tools so it's easier for developers to identify and set key accessibility properties.

Early investigations suggest the effort has 3 components. The first is adding an @beaninfo tag to the component source to identify the properties that are important to accessibility and would be exposed when introspection occurs. The next part is building a series of property editors that allow the setting of properties we want exposed in the builder then getting them included in dt.jar. The last aspect, which may turn out to not be needed, is to develop a customizer or series of customizers that essentially present the developer with a set of possible options or settings to choose from (e.g. associated with AccessibleState) which also gets placed into dt.jar.

There is no other solution out there for this "feature" to be better than. The point of this effort is to enable IDE tools and developers to do accessibility better.

---

### Implement the AccessibleEditableText interface on relevant text components

### ID: 4286284 Target of Opportunity

AccessibleText, an already implemented interface, provides the ability to find out information about text in a text component (e.g. font attribute, text content, font color, etc.) but it doesn't provide support enabling an assistive technology to fully interact with the text. What the AccessibleEditableText interface will do is provide the support that enables assistive

technologies to actually make changes in editable by the user text.

To be somewhat more specific the interface will allow an assistive technology to manipulate text in a number of ways. The following is an extraction of the methods this interface will provide. Please note that it's a first draft and we have changes/enhancements already planned.

```
public interface AccessibleEditableText extends AccessibleText {
 void setTextContents(String s);
 void insertTextAfterIndex(int part, int index);
 void insertTextBeforeIndex(int part, int index);
 void getTextRange(int startIndex, int endIndex);
 void delete(int startIndex, int endIndex);
 void cut(int startIndex, int endIndex);
 void paste(int startIndex);
 void replaceText(int startIndex, int endIndex, String s);
 void selectText(int startIndex, int endIndex); //Acts as a toggle
}
```

The benefit is greater user control of text thru the assistive technology (e.g. speech recognition) they are using.

---

### Implement full accessibility to HTML components

### ID: 4303259 Target of Opportunity

What is the feature and what problem is it solving?:
The new feature is to provide full accessibility to HTML components. Swing components such as JTextComponent can contain HTML components such as tables. Currently, these HTML components are not accessible to assistive technologies. To the extent that the HTML support in Swing becomes popular, we need to have strong accessible support for HTML compoents.

How will it work?:
The protected inner classes for Swing components that implement accessiblity will be modified to return accessibility information about HTML components contained by the Swing components.

What are the benefits over other solutions?:
The javax.accessibility package is the standard way for assistive technologies to obtain information about swing components.

---

### Implement AccessibleAction on components that support Swing Action

### ID: 4303272 Target of Opportunity

What is the feature and what problem is it solving?:
The new feature is to implement AccessibleAction on components (e.g., JTextComponent) that support Swing Actions. This feature will allow assistive technologies to perform such tasks as voice editing of text.

How will it work?:
The protected inner class that implements Accessibility for components will be modified to implement the AccessibleAction interface.

What are the benefits over other solutions?:

The AccessibleAction interface is the standard way for assistive technologies to obtain information about actions that a component supports.

---

### Implement discontinuous selection from the keyboard for list-like components

### ID: 4303294 Target of Opportunity

What is the feature and what problem is it solving?:
It is currently not possible to select discontinuous component from a list-like component. This significantly degrades access for disabled users must use a keyboard for an input device.

How will it work?:
Implement discontinuous selection in list-like components. This will probably require one or more API changes to Swing components.

What are the benefits over other solutions?:
(not applicable)

---

### Implement AccessibleJComboBox for the new JComboBox implementation

### ID: 4303297 Target of Opportunity

What is the feature and what problem is it solving?:
The Swing group has indicated that they will reimplement JComboBox. This will require that the protected inner class that implement accessibility for JComboBox needs to be modified to work with the new JComboBox implementation.

How will it work?:
The protected inner class that implements accessibility will be modified to work with the new JComboBox implementation.

What are the benefits over other solutions?
(not applicable)

---

# 7 Availability

---

### Add logging APIs to the Java platform.

### ID: 4286259 Group Driver

These APIs will be suitable for logging events from within the Java platform and from within Java applications.

This feature is intended to allow field service engineers to obtain information to help diagnose application problems in the field. This includes configuration issues and performance problems as well as application failures. The APIs are intended to be used from both within the Java platform implementation and from within Java applications to provide logging and tracing information.

Many operating systems provide some form of logging so that application programs can record key events, particularly on error conditions. It is desirable that the Java logging APIs should be able to interact with operating system logging services. In addition the logging APIs should enable logging to files or to network management services.

This API is being developed under the Java Community Process as JSR-047.

---

### Remote monitoring facility

### ID: 4288212 Target of Opportunity

Synopsis: Provide a remote monitoring facility which exposes a subset of the debugging and profiling information of a VM to trusted, authenticated clients.

Both the JVMDI and JVMPI (Java VM debugging and profiling interfaces) provide a wealth of information useful for optimizing Java applications and services. Critical operations such as application servers cannot use these facilities when they are running "hot", both because they can potentially expose sensitive information and because they can impact performance significantly.

We'd like to create a module to support remote monitoring, which exposes the subset of JVMDI and JVMPI information useful for monitoring live systems and which doesn't impact their performance significantly. This module would be loaded using the "-Xrun" parameter used to load profilers, and may be able to use JVMDI and JVMPI without change.

The remote monitoring module would provide a wire protocol as similar as possible to the Java Debug Wire Protocol (JDWP) to maximize code reuse by tool vendors. It would also use the subscription model of JVMPI, so that only requested information is sent to the client.

Secure access would be managed using Java encryption facilities -- I'm hoping that password creation and verification will have no impact on our export license. If it does, this module can be separately distributed if necessary (perhaps with some demo monitoring apps).

---

### provide APIs for failover & clustering

### ID: 4293539 Target of Opportunity

Many server systems support failover between servers, particularly within clusters. This tyically involves APIs for detecting and reporting partner failures.

Analogous APIs should be exposed in Java.

If this is approved for Merlin, a suitable JSR will be launched through the Java Community Process.

---

# 8 Deployment

---

### Plug-In Navigator OJI support

**ID: 4227202 Release Driver**

Netscape Navigator 6.0 will implement an Open Java Interface to allow the plug-in to provide the Java implementation in the browser. J2SE work is already well underway to support OJI and much of this support will ship in Kestrel.

We should continue this work and make sure that the Merlin version of the Plug-in fully supports OJI on win32, Linux, and Solaris.

---

### improve GUI feedback in plugin

**ID: 4290611 Group Driver**

To improve the usability of Java applets with Java Plug-in we should provide improved user feedback in three areas:

(1) We should avoid hanging the main browser thread while the plugin and VM load and initialize. Currently this leads to a fairly disruptive delay where page loading visibly freezes and the browser GUI stops responding. If we can load and initalize the VM asychronously, we can allow the browser to immediately paint the rest of the page, leading to a much better user experience.

(2) In each applet window we should provide a small progress bar or other GUI feedback while the VM is loading and initializing (including AWT initialization).

(3) In each applet window we should provide a small progress bar to report on applet download progress.

This applies to both IE and Navigator.

---

### additional APIs to support installers

**ID: 4290758 Group Driver**

We should add additional APIs to support installers written in the Java programming language. Some of these will also be of more general use.

The details need to be defined through the JCP process, but are likely to include:

- detailed versioning info for the Java 2 SDK
- detailed versioning info for the underlying OS and CPU
- APIs for finding free disk space

---

### Web Deployed Application Support

### ID: 4291051 Group Driver

The web deployed application project would allow users to deploy stand alone Java applications via a web browser. For large applications that don't need to be applets (i.e. that don't rely on browser services or integration with the enclosing HTML page), this feature would give the same deployment benefits that have heretofor only been available for applets.

The spec is being developed under the Java Community Process as JSR-056.

---

### Common DOM Support within Java Plug-in

### ID: 4291026 Target of Opportunity

Netscape and Internet Explorer use similar but different Document Object Models (DOM), making JavaScript integration harder to implement. This task would be to implement support for the new W3C standard, which is a superset of the 2 current competing DOMs. Assuming that both Microsoft & Netscape converge on the standard, this would allow for richer interaction between Java applets and the documents in the browser.

---

# 9 Virtual Machine

---

### Provide full 64 bit support

**ID: 4295833 Release Driver**

Merlin should provide full 64 bit support.

This includes making sure all native platform APIs (JNI, JVMDI, JVMPI, etc) are well defined at 64 bits, and scrubbing all shared, solaris, and win32 native source code to be 64 bit clean.

64 bit versions of the Java HotSpot(TM) VM implementation will be provided for
    Solaris on Sparc V9
    Solaris on IA-64
    Linux on IA-64
    Windows 2000 on IA-64
with emphasis on the Server system. The client system will eventually be adjusted as well, but the client system is not primarily targeting systems that require 64 bit addressing (that's why it is a client, after all). Thus primary focus will be on the server platform.

---

### Sharing between JVMs

**ID: 4287407 Group Driver**

Sharing these data structures will significantly reduce the VM-specific footprint and thus allow many more VMs to run simultaneously for a given amount of memory. We will investigate "skim-the-cream" solutions that will allow us sharing bytecodes across VMs.

---

### Reliability and Availability Improvements

**ID: 4287415 Group Driver**

Features for improved RAS (including better fault tolerance and better handling of low resources):

- improved GC performance (Scalability) (see: #4316400)
- better handling of low resource conditions (Reliability) (see: #4316406)
  (complete stack overflow handling, graceful out-of-memory handling)
- cleanup/redesign of JVMPI/JVMDI (Reliability, Accessability) (see: #4316402)
- improved fatal error reporting (Servicability)
  (provide symbolic info for crashes in native code,
  partial threads dump, VM state dump)
  (see: #4316406)
- "Browsable VM": will allow remote inspection of VM state via
  Web-Browser for performance debugging (Servicability, Scalability)
  (see: # (see: #4316408)

---

### One common source base for Hotspot JVM

**ID: 4294617 Group Driver**

For the next major release all versions for all platforms will be developped from a single consolidated source base (see a) below). However, individual source drops may still be required due to the way the bug fix / release process is working (see b) below.
In any case, the individual source drops will only differ in minor details due to late bug fixes. A comparison of the sources will reveal only minimal changes between different source drops for different platforms and will not hinder porting or understanding issues.

a) It makes a lot of sense to have a unified source base (workspace) for both client and server as well as all different platforms during development. The current workspace structure is already consolidated and used appropriately.

b) Towards the final releases, workspaces are forked off in order to facilitate the bug fixing and release process, and in order to avoid introducing bugs for one version when fixing bugs for another version. This is imperative, as any forced synchronization will make this process extremely slow and error prone. Since no major engineering is allowed during this phase, if individual work-spaces forked off from the consolidated workspace differ at all, they will do so only in minor details.

---

### Application control of thread stack size

**ID: 4294627 Group Driver**

Application control of thread stack size. We have run into this a lot; and controlling stack size individually is very important.

---

### improve JVM start up class loading

**ID: 4295838 Group Driver**

Faster Launch of the VM:

The start-up time of the VMis is dependent on the time used for class-loading. Improved class-loading speed will shorten start-up time of the VM.

---

### JVM interface cleanup

**ID: 4227211 Target of Opportunity**

Clean up and clearly document the JVM interface which can be seen as the contract between the VM and the libraries.

---

### Improve maintainability

**ID: 4287410 Target of Opportunity**

This task includes a system wide code review and removing leftover code for green thread implementation in JVM and

libraries. It will also involve general cleanups on the way.

---

**Reduce native code**

**ID: 4287412 Target of Opportunity**

Designing I/O libraries in Java:

Implementing the I/O libraries in Java requires an ExternalProxy abstraction for accessing memory outside the Java heap. In addition the compilers in HotSpot must inline the native functions in ExternalProxy to obtain adequate performance.

---

# 10 Internationalization

---

### Converter performance work

**ID: 4287463 Group Driver**

Improve the size, footprint, and speed performance of the existing encoding converters.

---

### Enable Thai locale support

**ID: 4287469 Group Driver**

Provide full Thai locale support, including support for tThai script layout.

This includes:
Thai locale data including country names, format strings, etc.
Calendar support including Thai Buddhost calendar
Complex Breakiterator support for Thai
Complex Collator support for Thai
Complex Text Layout support for Thai
Keyboard remapping for Thai

---

### Input Method Framework enhancements

**ID: 4287470 Group Driver**

1. The input method framework needs to be able to keep information about preferred input methods and interaction mechanisms (such as hot keys) between sessions.
RFE 4150589 - InputMethod switching menu should popup by some hotkey
RFE 4231842 - Sys-admin should be able to configure the system default IM
RFE 4248301 - Input method selection should persist between VM sessions

2. Better focus management
RFE 4293626 - Need support for focus-free windows

---

### Enable Hindi locale support

**ID: 4287473 Group Driver**

To fully enable Hindi locale support:
- Locale data including country names, format strings, etc.
- Calendar support
- ISCII character converters
- Add new fonts/glyphs
- US to Hindi keyboard remapping

---

### New I/O: Character-set API

**ID: 4313884 Group Driver**

An API for character-set support, including a service-provider interface

for pluggable converters. This API will give developers direct access the platform's built-in character-set converters and will also provide for the easy "plugging in" of new converters.

This API is being developed under the Java Community Process as part of JSR-051.

---

### Unicode 2.0 surrogate support

### ID: 4328816 Group Driver

Provide support for Unicode 2.0 surrogates.

---

### Support Unicode 3.0

### ID: 4287462 Target of Opportunity

This is to upgrade the Java 2 Platform to conform with the latest Unicode standard. Besides the new characters that have been added, some new properties, case mappings, and normalization rules have also been included. Some of the new features are:
- Add new character properties for the 10,000+ new characters. This means that several new properties will be created and existing ones will be updated in the Character class.
- Add new case mappings. Case mappings allow upper and lowercase operations.
- Update normalization tables. Unicode text takes many forms and mean essentially the same thing. Normalization is the process of reducing Unicode text to a common form that is easy to compare, collate, search, etc.
- Verify/Modify/Add Encoding Converters. Some converters may need modification to handle additional characters.
- Update bi-directional algorithm. This algorithm has been simplified in Unicode 3.0.
- Support surrogates. Although no surrogates are in the current standard, several thousand have been proposed.
- Update line break algorithms

---

### Character converter generator tool

### ID: 4287467 Target of Opportunity

This is to include the character converter tool as part of the J2SDK tool set. This tool has been used internally in the past, and will be updated for the new Character Converter SPI (as part of RFE 4287465).

---

### Keyboard configuration

### ID: 4287472 Target of Opportunity

Entering multilingual text in Java currently depends on the host having multiple keyboard layouts installed. Switching between these layouts is handled primarily by the host OS. Most users have never installed multiple keyboards and have only limited multilingual support. Additionally, developers often need detailed information about keyboards, their layouts, and about the position and syntactic meaning of keys. This API is not available or is deficient in the

current Java platform.

---

### Add UTF-7 to encoding converters

**ID: 4304013 Target of Opportunity**

Add UTF-7 to the standard set of supported encoding converters. This is useful when trying to pass Unicode data through transports that are only ASCII safe.

---

### getEncoding() Enhancements

**ID: 4304017 Target of Opportunity**

Supplement getEncoding() methods on such classes as InputStreamReader with methods to provide human readable, localizable, names. (getEncodingLocalized()?) Change the underlying converters to support this. The results of getEncoding() are used to create new encoding converters, so must map simply to those converters. However, the implementation should not assume that these strings are necessarily part of the converter class name, as is currently true for the sun.io package. The interfaces should be suitable to other converter implementations.

---

# 11 JDBC

---

### Bundle JDBC extension into core

**ID: 4290517 Group Driver**

At the moment we ship JDBC in two halves. Core JDBC functionality is delivered in the J2SE platform, but an unbundled extension delivers later APIs.

This separation is confusing to developers. I propose we bundle the entire JDBC standard extension into core, so that the entire JDBC API is available in the core J2SE platform.

This feature is dependent on 4290525: "Bundle JTA API"

---

### bundle parts of JTA API

**ID: 4290525 Group Driver**

JTA is the Java transaction API. It is currently delivered unbundled or as part of J2EE.

In order to support feature 4290517 "Bundle JDBC extension into core" it is also necessary to bundle parts of the JTA API.

We are not proposing to bundle JTS or any other specific transaction service provider, but only to bundle a sufficient subset of the JTA API to support JDBC.

---

### support disconnected rowsets

**ID: 4290543 Target of Opportunity**

Provide support for disconnected JDBC rowsets that can be retrieved in repsonse to a JDBC query, manipulated by a client, and then posted back en-masse to a server.

It is desirable that the transmission format should be an easily parsable open standard such as XML.

---

# 12 JNDI

---

### DNS service provider for JNDI

### ID: 4143066 Target of Opportunity

A DNS service provider for JNDI should be standard
part of the Java platform.
==========

A DNS provider would address the following issues:
1. Name resolution.

There are several high-priority bugs in classes_net filed against name resolution. The core of the problem is that the name resolution code relies on the platform native nameservice API to resolve names, and different platforms sometime have different nameservices. By providing a pure Java DNS client, for those applications that care, they can explicitly request the use of DNS and get around any platform dependencies. For now, the caching bugs in the networking code get around the problem by using an undocumented system property to control caching. The DNS service provider could be used by the InetAddress class, so that the InetAddress class is not dependent on platform-specific name resolution code. It could also be used by the InetAddress to do caching in a way compatible with the TTL settings on the DNS servers.

2. Locating LDAP servers. There is an IETF proposal (by Microsoft) to locate LDAP services using SRV records in the DNS. For Windows NT 2000, this is the way NT clients locate their LDAP/Active Directory servers. This allows clients to know "automatically" (i.e., without per-client administration) how to locate LDAP servers. Our LDAP client (the JNDI/LDAP service provider) currently must explicitly name the LDAP server. The ability to read SRV records from the DNS from a Java program (as provided by the DNS provider) would allow the LDAP service provider to automatically locate the LDAP service when such a feature is enabled in an environment such as an NT 2000 network.

3. Locating other services.
As NT 2000 becomes deployed and services use RFC 2052 and its variations to hold service address information, Java applications
that want to work using that model need access to DNS SRV records.

4. Mail address data. Allows Java applications, such as JavaMail,
to have access to mail address data (such as MX records) stored in the DNS.

5. Java client access to DNS data. Several external customers have requested this.

---

### Enhance and document JNDI service provider toolkit

### ID: 4283655 Target of Opportunity

The JNDI toolkit consists of classes and interfaces for helping developers write JNDI service providers. A couple of Sun's service providers (LDAP and NIS) currently use the toolkit.

Although most of the JNDI customers use the JNDI API and do not write

service providers themselves, with the increasing popularity of the J2EE platform, many customers writing application servers have requested assistance in developing service providers. The proposal is to enhance and document the existing toolkit, so that customers can use it to build their service providers.

Without this feature, the developer has to duplicate the code that's already in the toolkit (if it has obtained the source through the SCSL).

---

## DSML service provider for JNDI

### ID: 4283657 Target of Opportunity

Directory Services Markup Language (DSML) is a proposal to make directory data available in the XML format.
DSML enables XML-based applications in the following categories: eCommerce, eBusiness, network management, directory management, Intra-directory synchronization/access, and customer support.
See http://www.dsml.org for details on DSML.

The feature is to add a JNDI service provider for DSML access. This would enable applications to access directory data via DSML and allow customers to build Java-based applications that use DSML in the categories listed above.

---

## Support for GSS-SPNEGO/Kerberos V5 SASL mechanism

### ID: 4287808 Target of Opportunity

LDAP supports authentication via the Simple Authentication and Security Layer (SASL). See RFC 2222 for details on SASL.

NT2000's Active Directory supports the following SASL mechanisms: GSSAPI and GSS-SPNEGO. Currently, you can access Active Directory from the JNDI by using "simple" (clear text password) authentication. For real security, you need GSSAPI/GSS-SPNEGO with a Kerberos V5 plugin. SASL recommends that GSS-SPNEGO be used with SASL (instead of GSSAPI).

This proposal is to supply a GSS-SPNEGO/Kerberos V5 SASL mechanism that can be used with the LDAP provider. Depending on the progress of the Java SASL API (draft-weltman-java-sasl-02.txt) and its exportability status, this project may or may not use the Java SASL API. It might just use hardwired internal interfaces to get around export issues, and might not support encryption (security layer).

---

# 13 Core Libraries

---

### New I/O: Scalable I/O for sockets and files

### ID: 4313882 Release Driver

An API for scalable I/O operations on both files and sockets, in the form of either asynchronous requests or polling. This API will make it possible to write production-quality web and application servers that scale well to thousands of open connections and can easily take advantage of multiple processors.

This API is being developed under the Java Community Process as part of JSR-051.

---

### Preferences API/facility

### ID: 4286955 Group Driver

The proposed specification adds a simple API for managing user preference data and configuration data. Applications require preference and configuration data to adapt to different users, environments and needs. Applications need a way to store, retrieve, and modify this data.

Why the need isn't met by existing specifications - Currently, developers have two choices: either they make do without preference and configuration data (leading to reduced functionality), or they manage it in an ad hoc fashion. Often (though not always) preference and configuration data is stored in Properties Files, accessed through the java.util.Properties API. There are no standards as to where these files should reside on disk, or what they should be called. This makes it extremely difficult to backup a user's preference data, or transfer it from one machine to another. As the number of applications increases, the possibility of file name conflicts looms large. Further, this mechanism is of no help on platforms that lack a local disk, or where it is desirable that the data be stored in an external data store (such as an enterprise-wide LDAP directory service).

Less frequently, developers store user preference and configuration data in a directory service, accessed through the Java Naming and Directory Interface (JNDI) API. Unlike Properties, JNDI allows the use of arbitrary data stores (back-end neutrality). While JNDI is extremely powerful, it is also rather large, consisting of 5 packages and 83 classes. Further, JNDI does not provide any policy as to where in the directory name space the preference data should be stored, or in which name space.

In sum, neither Properties nor JNDI provide a simple, ubiquitous, back-end neutral preferences management facility. This JSR proposes such a facility, combining the simplicity of Properties with the back-end neutrality of JNDI. Further, it provides sufficient built-in policy to prevent name clashes, foster

consistency, and encourage robustness in the face of inaccessibility of the backing data store.

This API is being developed under the Java Community Process as JSR-010.

---

### Simple Assertion Facility

### ID: 4290640 Group Driver

Assertions are boolean expressions that the programmer believes to be true concerning the state of a computer program. For example, after sorting a list, the programmer might assert that the list is in ascending order. Evaluating assertions at runtime to confirm their validity is one of the most powerful tools for improving code quality, as it quickly uncovers the programmer's misconceptions concerning a program's behavior. Because there is a cost associated with checking assertions, it is highly desirable that it be possible to selectively disable assertion checking.

Many popular programming languages, including C and C++, have widely used and appreciated preprocessor-based assertion checking facilities. Several more recent programming languages, including Sather and Eiffel, have assertion-checking constructs as part of the language. The Java programming language was originally slated to contain such a construct, but due to time constraints, the construct never made it into the language.

JSR 41, at http://java.sun.com/aboutJava/communityprocess/jsr/jsr_041_asrt.html

---

### New I/O: Fast buffered binary and character I/O

### ID: 4313883 Group Driver

An API for fast buffered binary and character I/O, including the ability to map files into memory when that is supported by the underlying platform. This API will make it easier to write high-performance, I/O-intensive programs that manipulate streams or files of binary or character data.

This API is being developed under the Java Community Process as part of JSR-051.

---

### Cloneable doesn't define .clone

### ID: 4098033 Target of Opportunity

Cloneable doesn't define clone. This means that programmers cannot polymorphically clone objects, such as in:

```
for (int i = 0; i < myVector.size(); i++) {
    result.myVector.setElementAt(
        ((Cloneable) myVector.elementAt()).clone(), i);
}
```

Secondly, programmers have no idea whether clones are deep or shallow.

Since interfaces cannot be changed, suggested solution:
add an interface for deep clones:

```
public interface interface Copyable {
   public Object clone(); // guarantees deep clone
}
```

---

### Ability to update .jar files

### ID: 4227138 Target of Opportunity

At the moment jar files are read-only. To change the contents, one must rebuild the entire archive. The ability to add, update, or remove files from a jar file has been requested.

---

### support dense JAR format

### ID: 4290548 Target of Opportunity

In order to reduce download times for large applets or applications it would be desirable to support a denser JAR format.

As part of JRE 1.2.2 the "crunch" program was implemented as a special-case tool to compress rt.jar. This helped significantly to reduce the download size of the JRE. For large JARs the combination of "crunch" and a secondary compressor can achieve almost a 50% reduction over a normal compressed JAR file.

The "crunch" format may or may not be a useful beginning for this work. We need to define a well specified dense format that can be implemented by third pary tools, and we need to support that format in the JRE.

We do not need to change the Java VM or runtimes to support the new format. Instead it seems better to decompress a dense JAR into conventional JAR format as soon as it is downloaded and then use the regular JAR loader.

---

### Support chaining of exceptions

### ID: 4293532 Target of Opportunity

When one level of an application catches an exception, it often wants to report a different "higher level" exception upwards, but it also wants to include the full state of the lower level exception.

Some parts of J2SE already support this, e.g. the JDBC SQLException support chaining lower level exceptions.

A generic hook should be added to java.lang.Exception to support chaining of all kinds of exceptions.

---

### java.math.BigInteger prime generation performance improvement

**ID: 4294508 Target of Opportunity**

java.math.BigInteger is not used as often as it could be for cryptographic applications because the prime number generation is not fast enough. The tests as they stand are over conservative. It should be possible to make algorithmic improvements that will speed up the process significantly.

---

### java.math.BigDecimal string constructor speed improvement

**ID: 4294519 Target of Opportunity**

java.math.BigDecimal offers very good performance except in the case of the string constructor. The string constructor is unnecessarily wasteful and can be improved.

---

### New I/O: Scanning and formatting

**ID: 4313885 Target of Opportunity**

An API for text scanning (based upon regular expressions) and formatting (in the spirit of C's printf procedure). This API will bring regular expressions and a compact notation for formatted output to the Java platform, putting it on a par with other popular platforms such as Visual Basic and Perl.

This API is being developed under the Java Community Process as part of JSR-051.

---

### New I/O: Improved set of I/O exceptions

**ID: 4313886 Target of Opportunity**

A rich yet platform-independent set of I/O exceptions. This new set of exceptions will make it easier to write programs that recover from different types of I/O failures in different ways, and to write user interfaces that behave consistently on different platforms when I/O failures occur.

This API is being developed under the Java Community Process as part of JSR-051.

---

### New I/O: Improved filesystem interface

**ID: 4313887 Target of Opportunity**

A new filesystem interface that supports bulk access to file attributes (including MIME content types), escape to filesystem-specific APIs, and a service-provider interface for pluggable filesystem implementations. This API will work more consistently across platforms, will make it easier to write programs that gracefully handle the failure of filesystem operations, will provide more efficient access to a larger set of file attributes, will allow developers of sophisticated applications to take advantage of platform-specific features when absolutely necessary, and will allow

support for non-native filesystems, such as network filesystems, to be "plugged in" to the platform.

This API is being developed under the Java Community Process as part of JSR-051.

---

## Concise array literals

### ID: 4313888 Target of Opportunity

The Java programming language provides no convenient way to pass a variable number of arguments to a method, similar to the varargs feature of C or the &rest keyword of Common Lisp. This feature is extremely useful in GUI APIs, in constructing and manipulating collections, and in APIs for formatted output.

One of the goals of the New I/O project, for example, is to provide a simple API for formatted textual output similar to the well-known printf procedure of the standard C library. Developers have been asking for this feature for years; adding it will make Java that much more attractive an alternative to C, C++, and Visual Basic.

The central method of such an API accepts a format string followed by some arguments. The exact number of arguments required depends upon the content of the format string. Ideally we'd like to be able to write something like:

```
Formatter f = new Formatter(System.out);
f.fmt("%d bytes in %d seconds (%.2f KB/s)\n",
      nbytes, seconds, ((double)(nbytes / 1024) / (double)seconds));
```

The Java programming language, as presently defined, does not allow this.

If this feature is approved for Merlin then a JSR will be opened under the Java Community Process to define the appropriate, minimal, language change.

---

# 14 Networking

---

### IPv6 support

**ID: 4290596 Release Driver**

When the OS platform supports IP version 6, it should be accessible from the Java 2 Platform.

---

### URLEncoder and URLDecoder should support target character sets

**ID: 4257115 Target of Opportunity**

The java.net.URLEncoder and java.net.URLDecoder classes currently always encode and decode non-ascii characters using the platform's default character set encoding.

These two classes should be extended to allow encoding and decoding to and from specific character set encoding formats. The default character set should be UTF8.

---

### Socket Factory Support

**ID: 4290695 Target of Opportunity**

Add Socket Factory APIs to separate the action of choosing a socket and protocol type from the action of creating sockets. This is a simple framework that will enable flexible socket support in other Java 2 Platform and standard extension APIs and implementations and will allow for 3rd party socket extensions to be cleanly plugged in through socket factory handlers.

Among the problems this APIs addresses are: ablility to create unconnected sockets; create tunnelled sockets; have pluggable protocol family so as to support multiple protocols, for instance ICMP and RAW, which enable us and the users to create portable network management tools.

---

### Support for JNDI DNS service provider in InetAddress

**ID: 4290735 Target of Opportunity**

There are several high-priority bugs in classes_net filed against name resolution. The core of the problem is that the name resolution code relies on the platform native nameservice API to resolve names, and different platforms sometime have different nameservices. By providing a pure Java DNS client, for those applications that care, they can explicitly request the use of DNS and get around any platform dependencies.

For now, the caching bugs in the networking code get around the problem by using an undocumented system property to control caching. The DNS service provider could be used by the InetAddress class, so that the InetAddress class is not dependent on platform-specific name resolution code. It could also be used by the InetAddress to do caching in a way compatible with the TTL settings on the

DNS servers.

---

### Need support for FTP passive mode

### ID: 4304859 Target of Opportunity

RFC 959 File Transfer Protocol specifies that the DTP (Data Transfer Process), the process that establishes and manages the data connection, can be passive or active. The PASSIVE (PASV) command requests the server-DTP to "listen" on a data port (which is not its default data port) and to wait for a connection rather than initiate one upon receipt of a transfer command.

Currently, the FTPURLConnection class supports active FTP only. We would need passive FTP support in the presence of firewalls.

---

# 15 Security

---

### JSSE Optional Package in core.

### ID: 4286923 Release Driver

The Java Secure Socket Extension (JSSE) enables secure Internet communications. It implements a Java version of SSL (Secure Sockets Layer) and TLS (Transport Layer Security) protocols and includes functionality for data encryption, server authentication, message integrity, and optional client authentication. SSL and TLS are public-key-based network security protocols widely used to authenticate, privacy-protect, and ensure message integrity of transactions made over the Internet.

Using JSSE, developers can provide for the secure passage of data between a client and a server running any application protocol (such as HTTP, Telnet, NNTP, and FTP) over TCP/IP.

By abstracting the complex underlying security algorithms and "handshaking" mechanisms, JSSE minimizes the risk of creating dangerous but subtle security vulnerabilities. Furthermore, it simplifies application development by serving as a building block which developers can integrate directly into their applications.

JSSE is implemented entirely in Java and
is currently being delivered as an optional package at
http://java.sun.com/products/jsse/

---

### JAAS Optional Package in core.

### ID: 4286906 Group Driver

The latest release of the Java(TM) 2 Software Development Kit, v 1.3, provides a means to enforce access controls based on where code came from and who signed it. The need for such access controls derives from the distributed nature of the Java(TM) platform, where, for instance, a remote applet may be downloaded over a public network and then run locally.

The Java 2 platform, however, lacks the means to enforce similar access controls based on who runs the code.
To provide this type of access control, the Java(TM) 2 security architecture requires additional support for authentication (determining who's actually running the code), and extensions to the existing authorization components to enforce new access controls based on who was authenticated.
The Java(TM) Authentication and Authorization Service (JAAS) framework augments the Java 2 platform with such support.

The JAAS authentication framework is based on
PAM (Pluggable Authentication Modules),
and therefore supports an architecture that
allows system administrators to plug in the appropriate authentication services to meet their security requirements.

The architecture also enables applications to remain independent from the underlying authentication services. Hence as new authentication services become available or as current services are updated, system administrators can easily plug them in without having to modify or recompile existing applications.

The JAAS access control framework extends the Java 2 access control architecture and security policy in a compatible fashion, and doesn't require modifications to the SecurityManager. Therefore, existing subclass SecurityManager implementations will continue to function seamlessly.

The JAAS framework is implemented entirely in Java and is currently being delivered as an optional package at http://java.sun.com/products/jaas/

---

## JCE Optional Package in core.

### ID: 4287162 Group Driver

The Java Cryptography Extension (JCE) 1.2 is an officially released Standard Extension to the Java 2 Platform. JCE 1.2 provides a framework and implementation for encryption, key generation and key agreement, and Message Authentication Code (MAC) algorithms. JCE 1.2 supplements the Java 2 platform, which already includes interfaces and implementations of message digests and digital signatures.

But JCE 1.2 cannot be exported outside the U.S. or Canada. This has limited its usability and deployment. JCE Fontanel (the next release of JCE) has made JCE exportable since its added JCE framework and provider mutual authentication and a mechanism to control available cryptographic strength via jurisdiction policy files. This is very important to keep and attract JCE users, and ensure ubiquity. This also makes it possible to add JCE into the core Java SDK.

Compared to more mature cryptography frameworks (such as CDSA), JCE 1.2 lacks key wrapping and some key management functionality such as key usage control. Without key wrapping, exporting and importing keys is difficult. JCE Fontanel supports key wrapping and key usage control.

The APIs in JCE would have be in the core SDK except for the export issue. The JCE should be added to the core SDK now that the export issue has been resolved.

JCE Fontanel will be delivered as an Optional Package before the Merlin release.

==========

---

## Performance tuning: Improve Performance w/SecurityManager

### ID: 4227199 Target of Opportunity

The Java 2 security model offers fined grained access control which can impact runtime performance. Currently the only real optimization made is in the removal of null and duplicate adjacent protection domains. We currently do not have sufficient metrics to know whether this optimization is worth the effort or whether there are

heuristics which we can use to add feedback to the optimization algorithm.

Other forms of optimizations may also be possible. For example, it may be advantageous to construct the contents of PermissionCollections in the order of most frequently used Permission. Additionally, the backing store to most if not all PermissionCollections is typically a Hashtable. When instantiating the Hashtable we rely on default load and capacity settings which may be suboptimal. Using a more appropriate collection class could improve performance as well as supply better synchronization semantics.

---

### PKCS (Public-Key Cryptography Standards) support.

### ID: 4286953 Target of Opportunity

PKCS is a set of public-key cryptography standards developed by RSA Laboratories (http://www.rsasecurity.com/rsalabs/pkcs). The PKCS standards cover RSA encryption, Diffie-Hellman key agreement, password-based encryption, extended-certificate syntax, cryptographic message syntax, private-key information syntax, and certification request syntax, as well as selected attributes.

PKCS is used in many security standards and products today, such as S/MIME and PKIX (RFC 2459). This feature is to define a set of Java APIs and implementation for the PKCS #1, #5, #7, #8, #9, #10, #12 standards.

---

### Certificate path building and verification.

### ID: 4287221 Target of Opportunity

The Certification Path API is a Java API for building and verifying certification paths (also known as certificate chains). A certification path is an ordered list of certificates. If a certification path meets certain validation rules, it may be used to securely establish the mapping of a public key to a name.

A reference implementation (currently under development) will provide a PKIX-compliant (RFC 2459) implementation, including support for fetching certificates from an LDAP directory via JNDI and checking revocation using CRLs. This API will make life easier for code that needs to build or verify certification paths (such as SSL code, signed code verification, S/MIME code, etc.).

See http://java.sun.com/aboutJava/communityprocess/jsr/jsr_055_certp.html for more information.

---

### Kerberos via J-GSS

### ID: 4287239 Target of Opportunity

At the outset, it is important to understand the GSS-API operational paradigm. GSS-API is not responsible for establishing communication channels between the client and server application. Instead, it is the

GSS-API caller's responsibility to establish a communication link with a peer application. Typically, when
an application uses GSS-API, it has access to a local GSS-API implementation. The local GSS-API implementation hides beneath it one or more security mechanisms. Thanks to GSS-API, the security mechanisms are transparent to the application. In fact, it is possible to plug-in a totally different security mechanism without affecting the application.

GSS-API requires a security mechcanism. Kerberos - an authentication protocol based on Needham and Schroeder secret-key protocol - became well-established in many security environments.

==========

The Java 2 Platform must have a way to allow applications to utilize the Kerberos infrastructure that the OS might have access to. The standard way of accessing the authentication, privacy and integrity services of Kerberos is via the Generic Security Services API (GSS-API) defined by RFC 2743. The Java bindings of this API (draft-ietf-cat-gssv2-javabind-05.txt) has been forwarded to the IESG for consideration as a proposed standard by the Common Authentication Technology working group of the IETF. The Java 2 Platform must include an implementation of this API with Kerberos support underneath.

---

# 16 RMI

### Custom remote references

### ID: 4227190 Group Driver

Communication with remote objects has different requirements; it may need to use a different transport, protocol, or pass additional information in a remote method call. Use of the current RMI transport and protocol with only slight modifications is also useful.

To support differing communication requirements, we'd like to expose "custom remote reference types" in RMI. In RMI, a remote reference type controls the communication between client and the remote object. There are several remote reference types that exist in RMI today: a simple "unicast" reference type, a reference type that makes use of custom socket factories, and a reference type for an "activatable" remote object. Currently, only the private RMI implementation can make use of custom remote reference types. Since customizing reference types is useful in general, we'd like to add this feature to the public RMI API. That is, add public APIs so that a remote object can specify the custom remote reference type used for remote communication from clients to that remote object.

This feature enables secure communication, group communication, non TCP-based communication, and other custom remote communication to be modeled as remote method calls. This feature completely generalizes the RMI framework to support a wide variety of distributed communication under a single API.

As part of this work, we need to define public protocol and transport-level APIs so that applications can implement custom remote reference types without having to re-implement the entire RMI transport and protocol. Allowing custom reference types without any support for transport and protocol management would place too much burden on the server programmer that used such references (meaning that they would essentially have to reimplement all of the RMI internals).

As part of this work, an enhanced RMI protocol will be added to address certain performance and scalability issues. Along with this protocol work we will support asynchronous calls, a lower level distributed garbage collection mechanism for increased performance, call cancellation/

timeout, server-specified thread control for
remote calls, as well as increased performance
by implementing a caching scheme for serialization.

---

### Codebase map

### ID: 4227196 Group Driver

RMI users need a hook into the mechanism that RMI uses to obtain an external code-base annotation for outgoing classes. RMI developers need more control of the codebase annotation in order to solve the following problems:

1. Need ability to separate import and export codebases. If implementation classes are loaded through an RMIClassLoader (which is the case for activatable objects), then an RMI client will always be able download any classes in the codebase annotation (this includes client *and* implementation classes). Service providers do not have a convenient way to prevent clients from ever being able to access server code - this can be a problem for proprietary systems which wish to protect intellectual property. For more information about this problem, see 4109372.

2. RMI should support refresh of the codebase property:

The mechanism for making more granular selection of codebase annotations should take into account the need to refresh the codebase property.

3. Currently all classes sent from the same VM or RMIClassLoader have to be annotated with the same codebase (even sets of classes that are logically unrelated). This restriction makes it possible for VMs which are receiving downloaded code to make use of mistaken or inadvertent dependencies on classes in codebase annotations that contain multiple "logical" groups of classes. If codebases could be constructed which only contained the minimal set of classes in a "logial" group, client applications would be forced to use server classes in a more robust manner.

4. It should be possible for a single VM to host multiple logical services that each have their own distinct codebase annotation. A codebase map file may be able to help differentiate codebases of different services.

---

### Granularity of connections to threads

### ID: 4295840 Group Driver

The RMI performance issue with granularity of connections to threads needs to be fixed

---

### RMI security framework

### ID: 4302502 Group Driver

We'd like to extend the security architecture to distributed systems built using RMI, by providing mechanisms to mutually authenticate client and server subjects during a remote call, protect the communication from third parties, and execute code in the server on behalf of the client's subject.

The proposed API is intentionally at a very high level; cryptographic mechanisms and protocols are not exposed, so that code written to the API is more portable. An underlying service provider interface will allow specific mechanisms and protocols to be configured into the framework.

Although this is an addition to RMI, the API is designed to be applicable more generally to remote services that are defined in terms of interfaces. Specifically, the API is also targeted for direct use in Jini(TM) connection technology. As such, it is designed for use with general front-end proxy objects for remote services, not just for use with RMI stubs.

The main components of the API are as follows:

An extensible set of constraint classes is provided, allowing both clients and servers to express security constraints to be applied to remote calls. The basic constraints cover communication integrity, client and server authentication, and delegation. Support for confidentiality may be added if it can be done in a way that does not restrict export classification of the SDK. Constraints come in two basic flavors: requirements and preferences. A requirement is a mandatory constraint that must be satisfied for the remote call. A preference is a desired constraint, to be satisfied if possible. A server can specify different constraints for each remote method. Clients can attach constraints directly to a proxy, in which case they apply to all remote calls made through that proxy by any thread, or clients can specify constraints contextually, in which case they apply to all secure remote calls made anywhere within that scope by that thread.

A general mechanism is provided for clients to verify basic trust that a downloaded proxy object (that may be an instance of a dynamically downloaded class) will correctly obey the client's security constraints.

Secure versions of unicast and activatable remote objects are provided.

Secure versions of the RMI registry and activation system daemon are provided.

Support is provided for configuration files, for end users to specify the security constraints to use when remote objects are exported.

Service provider layer is defined, so that new security mechanisms and protocols can be plugged in.

A specific provider that uses JSSE will be included, if it can be done in a way that does not restrict export classification of the SDK, and if JSSE is included in the SDK.

---

# 17 CORBA

---

### IDL+RIP POA

### ID: 4227147 Release Driver

Implementation of the POA, Server activation, and persistent objects.

---

### Portable Interceptors APIs in Merlin

### ID: 4307834 Group Driver

```
Bugtraq-#   :  4307834  (Portable Interceptors APIs)
Desc        :  Portable Interceptors APIs in Merlin
```

The purpose of this feature is to provide a set of OMG standard APIs, that can be used by J2EE Reference Implementation (RI) or other J2EE vendors during development.

The current RI implementation uses private APIs into the ORB for the following:

(a). set/get information using private interception points for security security context.
(b). setting codebase during object creation in the object reference.

The Portable Interceptors provide hooks for the Application Programmers to write their own code, that can be invoked at various points in the ORB. The inclusion of Portable Interceptors APIs will enable:

(a). the J2EE RI (Reference Implementation) to use the complete ORB implementation from the JDK core. Currently the ORB implementation is bundled with the J2EE Reference Implementation. Being able to use the ORB implementation from JDK core helps reduce the J2EE RI footprint.

(b). Using the OMG standard APIs, the J2EE RI will be able to run on any vendor's ORB.

(c). Any other J2EE vendor can develop their code using the complete ORB provided by the JDK core.

The inclusion of this feature as standard OMG APIs is dependent on the Portable Interceptors FTF (Finalization Task Force) and OMG propose/dispose process. We believe that this process will complete before shipping Merlin. If the OMG process is delayed, we have a backup plan to not ship them as standard APIs.

The Portable Interceptor APIs are defined by OMG Specification: orbos/99-12-02
The errata is specified by doc-#: orbos/00-01-01

---

### Persistent CosNaming Implementation

### ID: 4227145 Target of Opportunity

Add an implementation of CosNaming that stores its naming database persistently. IDL currently has a transient implementation of this API. The implementation will

```
require the POA.
```

---

# 18 Tools

---

### JPDA: Class File Redefinition

**ID: 4287595 Target of Opportunity**

This feature encapsulates the ability to subsitute modified code in a running application through the debugger APIs. For example, one can recompile a single class and replace the old instance with the new instance. This minimizes time spent by the developer to make small, incremental changes to their application.

This feature is required by some tools vendors before they move to JPDA. They have created hooks in their specialized VM and specialized debugger interfaces for this functionality.

---

### Rewrite javah as a JavaDoc doclet

**ID: 4287641 Target of Opportunity**

Rewrite javah utility as a doclet. This will be necessary once the old compiler is retired, as the current javah has dependencies on the old compiler.

---

### Rewrite javap as a JavaDoc doclet

**ID: 4287648 Target of Opportunity**

Rewrite javap as a JavaDoc doclet. Doing this task will remove the dependency on the old compiler, as well as unifying the static tools.

---

### Member categorization

**ID: 4287701 Target of Opportunity**

This JavaDoc feature will allow one to group class members together with related functionality for documentation reasons.

---

# 19 Miscellaneous

---

### Archiving Graphs of JavaBeans

**ID: 4290987 Group Driver**

This project is to deliver an alternative persistance mechanism for archiving graphs of JavaBeans. The goal is to provide an efficient and resilient archiving mechanism that is suitable for use in IDEs and will make archiving and recovery of GUI designs more reliable and portable.

This API is being developed under the Java Community Process as JSR-057.

---

### Java platform messages should be unique

**ID: 4293473 Group Driver**

Currently the Java runtimes sometimes generate the same message from different places in slightly different circumstances. This can make it hard to identify the actual source of a given problem.

We should ensure that messages originating from differnet places in the runtimes should be unique. So given a message string it should be possibly to uniquely identify its origin and meaning.

---

### Make the Java sound "C" code portable by adding 64-bit support

**ID: 4303630 Group Driver**

The Java Sound "C" code is already quite portable, but needs work for the new 64-bit OS's. Also, the C++ comments need to be changed to C comment style.

---

### Remove old demos that aren't valuable examples

**ID: 4288159 Target of Opportunity**

Remove some of the old demos that aren't valuable examples of Java programming. Analysis of the current demos will need to be done to see what's valuable. This does not include demos such as the 2D or Swing demos; just the base Java demos.

---

### Add new demos that are valuable examples

**ID: 4288161 Target of Opportunity**

Possibly create 2 or 3 very worthwhile demos of Java programming. Java Applets as well as Java applications that showcase good coding practices and is a somewhat functional demonstration of Java.

---

### #else and #endif in code should have comments

### ID: 4294635 Target of Opportunity

#else and #endif in code should have comments. Often, we put our code around #ifdef #endif combination. If Sun modifies the code in the same place and provide #ifdef's they clash. And the automatic merge tool picks one of the 2 #endif's only. If we put specific comments after #else and #endif, they will be different lines and not considered the same.

Thus, use

```
#if defined(__linux__)
<code for linux>
#else /* __linux__ */
<code for solaris>
#endif /* __linux__ */
```

---

# 20 Feature List Change History

## 20.1 Changes between 0.20 and 0.30:

Miscellaneous book-keeping changes:

- 4304014 “Extend URLEncoder” is replaced with 4257115 “URLEncoder and URLDecoder should support target character sets.” The two features solve the same problem, and the change is just internal book-keeping.
- 4293386 “Full 64 bit support” has been merged with 4295833 “Provide 64 bit support.” Both features were release drivers and they specified similar functionality, so they have been merged as one feature. There is no substantive change!
- 4289849 “Improved scrolling” has been merged with 4285182 “Improved Scrolling”. These two features were essentially duplicates, one being an AWT perspective and the other being a Swing perspective on the same performance work, so it makes more sense to track them as a single feature.

Features added:

- 4307834 “CORBA Portable Interceptors.” This had been accidentally omitted from earlier drafts.
- 4290704 “Swing getListener project completion.” This had been targeted for Tiger but now appears doable for Merlin.

Targets of Opportunity removed:

- 4304015 “Extracting META HTTP data.” The team reported they would not reach this.
- 4290699 “Repaint code from Swing into AWT.” After investigation the team concluded that this would provide little performance benefit and might introduce incompatibilities.

# 21 Index by Feature ID