

ICEBROWSER® / ICEREADER® SDK

ICEbrowser® and ICEREADER® SDK offer a set of customizable and extendible browser components for developers of software systems that need to render web applications. These SDKs support the latest Internet standards and protocols, empowering you to build highly customized client solutions.

Both SDKs use Java technology, which significantly reduces development costs in porting, releasing, and supporting sophisticated Web applications across heterogeneous environments. Their modular architectures, compliance with current Web standards, and rich set of development APIs make these web browser SDKs the most versatile development kits in the Java world.

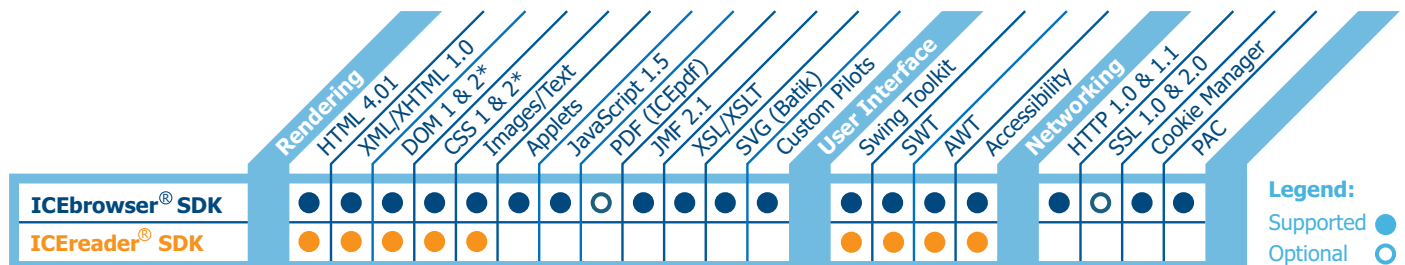
At a static code-base size of 650KB–1.5MB (depending on configuration), you can create standalone Web clients or web clients embedded within larger software systems. Enterprise web applications will be rendered accurately and securely using ICESoft's proprietary Java browser engine.

ICEbrowser SDK offers robust HTML/XML page rendering and delivers a rich user experience by supporting JavaScript, applets, and other types of web content. ICEREADER SDK includes the subset of Java libraries found in the ICEbrowser SDK typically used for rendering static web content.

The provided sample GUIs, for both AWT and JFC/Swing, are fully customizable, allowing you to extend or constrain end-user functionality, and to control the application's look and feel.

These SDKs also provide numerous examples and complete documentation.

Supported Standards



* ICESoft implements the commonly used specifications. For more information, see the ICEbrowser/ICEREADER SDK Developer's Guide.

Enterprise Deployment Scenarios

- > **Thick clients:** These modular, Java-based SDKs allow you to readily embed the functional web components you require within thick, specialized web-enabled applications.
- > **Thin clients:** ICEbrowser and ICEREADER SDKs can be used as a standalone application that can render secure, dynamic web content in applications which require JavaScript and Applet support. Thin client implementations are often married with Java Web Start technology, enabling centralized application management and deployment.
- > **Browser Engine:** The ICEbrowser and ICEREADER SDKs are ideal for integrated applications that require functionality such as DOM tree construction and manipulation, or web content crawling and extraction.
- > **Headless Server:** Both ICEbrowser and ICEREADER SDKs support rendering documents in non-visual and headless Java environments. These Java browser SDKs are also commonly used in J2EE environments as a browser engine to output parsed web content to a specified image or file format.

Supported Platforms



The SDKs are 100% Java and capable of running on the following JVMs:

Windows:

- Sun JDK 1.3.1
- Sun J2SE 1.4.x
- Sun J2SE 1.5.0

Linux:

- Sun JDK 1.3.1
- Sun J2SE 1.4.x
- Sun J2SE 1.5.0

Solaris:

- Sun JDK 1.3.1
- Sun J2SE 1.4.x
- Sun J2SE 1.5.0

Mac OS/X:

- Apple JDK 1.3.1
- Apple J2SE 1.4.x

ICEbrowser Architecture

The ICEbrowser architecture is modular, flexible, and versatile. The combination of Pilots, Scripters, and the ICEbrowser core provides the basis for advanced HTML rendering and web browsing, and a foundation for custom application development.

Pilots

A pilot is a rendering module for a specific MIME type. Pilots are similar to the plug-ins of traditional browsers, but are compact, platform independent, and can be loaded dynamically.

HTML/XML Pilot

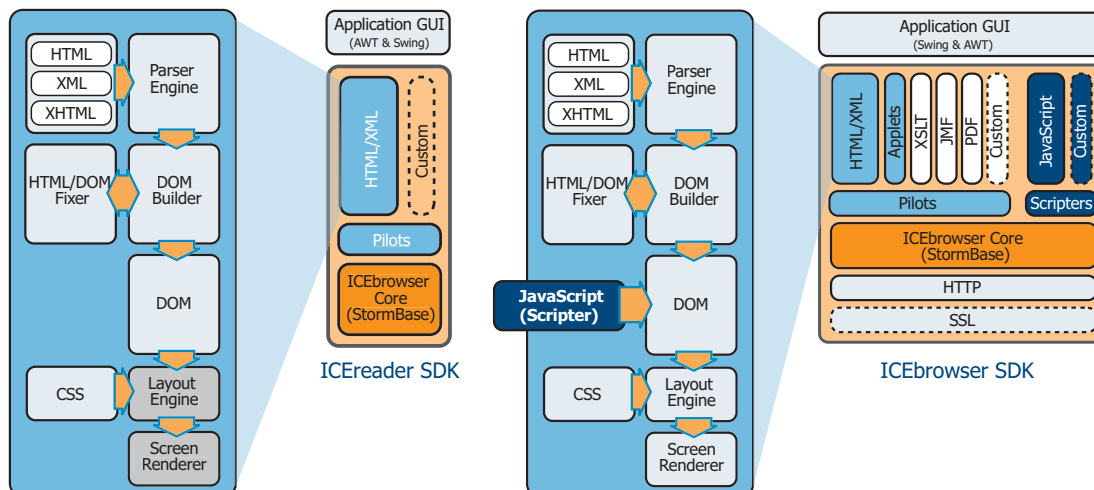
The HTML/XML Pilot is the rendering module that reads in HTML/XML source, manipulates it as necessary, and displays the content to a screen. Distinct components within the pilot perform each task.

- > The Parser Engine analyzes the source and creates a data tree that represents the tags in the source and their hierarchy.
- > The DOM Builder builds the Document Object Model (DOM). As part of this process, the DOM Builder calls the HTML/DOM Fixer to validate and fix the source. Invalid HTML is manipulated using fuzzy logic to make it valid.
- > Once a valid DOM is built, the Layout Engine organizes the elements as specified by the rules of the Cascading Style Sheet (CSS).
- > Finally, the Screen Renderer renders the representation of the DOM and CSS to a display screen.

Additional Pilots

An ICEbrowser implementation can contain additional pilots for other MIME types, such as XSLT, JMF, and PDF. The Applet Pilot, for example, allows ICEbrowser to render Java applets. Some pilots require a third-party library. The XSLT Pilot, for example, requires the Xalan-Java JAR files. Custom pilots can also be built and integrated.

Architecture Diagrams





Scripters

The Scripter is the module that provides the “glue” between the HTML/XML pilot and the ICEbrowser core. The JavaScript Scripter interprets JavaScript commands to provide advanced HTML rendering capabilities. Custom scripters can be built and integrated to allow commands to be run in other scripting languages. Pilots can then expose their methods into the scripting environment without depending on a particular scripter.

ICEbrowser Core

The ICEbrowser core (StormBase) is the integrating layer that provides the framework for the pilots and scripters. Through the core, pilots interact with their host environment and can call and interact with other pilots and scripters.

HTTP and SSL Modules

The HTTP module is a proprietary implementation of the HTTP/1.0 and HTTP/1.1 protocols. When combined with SSL, it extends support to HTTPS. The optional ICEssl module supports secure networking over the Internet with both the SSL 2.0 and SSL 3.0 protocols.

Application GUI

ICEbrowser has a fully customizable GUI. Developers can constrain or extend user functionality, create browser skins, or develop a specialized user interface using Swing or AWT.

SDK API Summary

> These APIs are not available with the Bean products.

Legend:

- Supported ●
- Optional ○

		ICEbrowser	ICereader
StormBase			
Life cycle	Create/dispose/find viewports, get/set active viewport, enable/disable pilots, scripters	●	●
Navigation	Render content, stop loading, reload, go forward, go back	●	●
History Mgmt.	Access document viewing history, canGoForward, canGoBack	●	●
Viewport Status	Receive notifications for all viewport status changes (loading status/ progress, content metadata, errors)	●	●
Pilot/Scripter Factories	Access configured pilots and scripter (JavaScript) information	●	
Scripter Callback	Define JavaScript behaviors (dialog windows, longRunningScript, allowScriptClose, errors/warnings, etc.)	●	
Custom Pilots	Configure custom rendering pilots for new content types	●	
HTML 4 Pilots			
Pilot	Configure and manage HTML4 pilot (DOM, component hierarchy, encoding, selected text, page info, etc.)	●	●
View	Access HTML view information (size, location, DOM node bounding-box, FocusManager, scroller, toolkits, zoom)	●	●
DOM	Access and manipulate document DOM dynamically from Java	●	●
CSS	Access and manipulate CSS rules	●	●
Search	Perform full-text searches of document text (case insensitive, whole-word, wrap)	●	●
ActionHandler	Override/extend to alter default browser actions for UI events (keyboard, mouse DOM events)	●	●
Accessibility	Supports javax accessibility, AccessibleText, AccessibleHyperText and JAWS screen-reader APIs	●	●
ComponentFactory	Support multiple toolkits for HTML form widgets (AWT, Swing, custom)	●	●
Font Provider	Configure and manage multiple font providers, custom font providers	●	●
Printing			
Page Format	Control page setup for printing/print-preview (paper size, orientation, margins, scale, header/footer, framesets, etc.)	●	●
Print Dialogs	Display Page Setup, Print Preview dialogs (AWT & Swing)	●	●
Print	Display printer dialog, print entire document or selected pages	●	●
Utilities			
Emulation	Access/set the current browser emulation mode (MSIE, NS6/Mozilla, ICEbrowser)	●	●
Memory Mgmt.	Configure and manage active memory management for platforms with restricted Java heap size	●	●
Debug	Configure various levels of debug and trace output to assist with problem resolution	●	●
HTTP/HTTPS			
Connection Mgmt.	Configure HTTP/S connection settings	●	
Authentication Mgmt.	Configure and manage HTTP authentications (Basic, Digest, and NTLM authentications)	●	
Cache Mgmt.	Configure and manage HTTP caching (memory cache, disk cache, custom cache)	●	
Cookie Mgmt.	Configure and manage HTTP cookies	●	
Proxy Mgmt.	Configure HTTP and HTTPS proxy host and port, PAC (Proxy Auto Config) proxies	●	
Connection Listener	Receive notifications/ intercept all requests and responses (examine, modify, and cancel HTTP transactions)	●	
ICEssl			
Certificate Mgmt.	Manage SSL certificates (CAList, server certificates, client certificates, signlists, etc.)	○	

Note: For more information, see the ICEbrowser/ICereader SDK Developer's Guide.

SDK Benefits Summary



Industry-Leading Performance
World's fastest pure-Java browser with best-in-class W3C standards support
Robust, proven HTML fixer for non-standard, real-world HTML
Responsive DHTML
Enhanced performance HTTP/HTTPS protocol support
An embeddable Java component that provides a native browser user experience
Over 10 million deployments in 35 countries
General
Internationalization support (Unicode support, left-to-right and right-to-left layout)
Extensible plugin architecture (ICEbrowser SDK only)
History management
Full support for headless and non-visual platforms (render to image, render to PDF, render to printer, etc.)
Exclusive Internet Explorer (MSIE) emulation mode supports many MSIE-proprietary features
User Interface
AWT, Swing, and SWT* toolkit support (*requires AWT/SWT Bridge)
Document scaling/zooming
Text selection
Autoscrolls document while selecting text via mouse drag
System clipboard support
Text searching
Configurable scrolling and scrollbar behavior
Mouse scroll-wheel support
Automatic mouse cursor/pointer support
Keyboard support consistent with native browsers
Anti-aliased text
Robust frameset and frames support for all features (search, selection, etc.)
Visited links highlighting
Tooltips
Fully customizable JavaScript dialogs and behavior via callback architecture (ICEbrowser SDK only)
Configurable font support, including custom font providers
Printing
Supports both JDK 1.1.8 and Java 2 print APIs
Automated layout to page size and orientation, or user-defined
Accurate printing of background colors and images
Full support for margins, headers, footers, and watermarks
Print entire document or user-specified page ranges
Print framesets "As laid out on screen" or "Each frame individually"
"Smart Page Break" technology for optimized pagination
Professional Page Setup and Print Preview dialogs
Accessibility
Supports Java Accessibility API
Supports industry-leading JAWS screen-reader technology
HTTP/HTTPS
HTTP 1.0 and 1.1 support
Authentication Manager <ul style="list-style-type: none">-Basic, Digest, and NTLM* authentication (*optional)
Configurable Cookie Manager
Configurable Cache Manager (MemoryCache/DiskCache/Custom)
Configurable Proxy Manager <ul style="list-style-type: none">- Supports HTTP, HTTPS, and PAC (Proxy Auto Config) proxy configurations
HTTP Event Listener support (intercept HTTP transactions, form-posts, etc.)
SSL support (optional) <ul style="list-style-type: none">- Certificate Manager- Customize SSL dialogs, certificate management via callback architecture
Supports real-world, non-standards compliant web-servers
Faster and more robust than standard JDK HTTP implementation
Small Footprint
Static footprint as low as 690 KB
Active memory management
Deployment
Java applet - standard Java applet security sandbox, or extended privileges via signed applet
Java Web Start (JNLP)
Executable .jar, and standalone application