BD-J, or **Blu-ray Disc Java**, is a specification supporting Java ME (specifically the Personal Basis Profile of the Connected Device Configuration or CDC) Xlets for advanced content on Blu-ray Disc and the Packaged Media profile of Globally Executable MHP (GEM).

BD-J allows bonus content on Blu-ray Disc titles to be far more sophisticated than bonus content provided by standard DVD, including network access, picture-in-picture and access to expanded local storage. Collectively, these features (other than internet access) are referred to as "**Bonus View**", and the addition of internet access is called "**BD Live**". BD-J was developed by the Blu-ray Disc Association. All Blu-ray Disc players supporting video content are required by the specification to support BD-J.[1] Starting on October 31, 2007, all *new* players are required to have hardware support for the "Bonus View" features, but the players may require future firmware updates to enable the features. "BD Live" support is always optional for a BD player.[2]

Sony's PlayStation 3 has been the *de facto* leader in compliance and support of BD-J. The PlayStation 3 added Blu-ray Profile 1.1 support with a firmware upgrade and was used to showcase BD-Live at CES 2008 in January.

Contents

BD-J Xlet capabilities

- The invocation of BD-J Xlets are triggered by events occurring around them—for example, by the selection of a film title, or by the insertion of a new disc. Xlets in turn can then call other Xlets into play.
- Security in BD-J is based on the Java platform security model. That is, signed applications in JARs can perform more tasks than a non-signed, such as Read/Write access to local storage, network access, selection of other titles on the BD-ROM disc, and control of other running BD-J applications.
- Xlets (as part of the CDC Personal Basis Profile) have no GUI (i.e. no AWT widgets such as `java.awt.Button`), so additional classes are called into play for generating animation and GUI. The BD-J uses the Havi UI device model and widget set for remote control use, but it is extended to allow for the BD supported resolutions and BD supported A/V controls.
- BD-J has classes that allow the user to synchronize accurately to specific frames in the film.
- There are two types of video synchronizations allowed, one called "loose synchronization", which uses a call back method and is accurate to within several frames of the event, and the other being "tight synchronization", which uses the package `org.bluray`. Tight synchronization allows applications to synchronize accurately to the exact frame using timecodes from the package `javax.media.Time` of JMF (Java Media Framework).
- A BD-J application's GUI can be operated with a remote control with a required set of keys and an optional pointing device. The set of required keys includes at least the keys needed to support the

User Operations in HDMV applications.

- The GUI framework in BD-J includes the HAVi(6) UI framework mandated by GEM; it is not a desktop GUI framework like Swing or AWT. The GUI framework is based on the core of AWT as specified by PBP, but the widget set includes mechanisms for remote control navigation from GEM and easy customization of look and feel from HAVi.
- BD-J includes a media framework similar to JMF for the playback of media content related to the BD-ROM disc. It is assumed that the BD-ROM disc will be the prime source for media files, but it will not be the only one; other sources could be the studio's web server and local storage.
- BD-J includes standard Java libraries for decoding and displaying images in JFIF (JPEG), PNG and other image formats. These images can be displayed on the Java graphics plane using standard Java graphics functions. An image can also be rendered in the background plane using a BD-J specific package.
- Text can be rendered using standard Java text functions. These text-rendering functions are extended with a more advanced text layout manager that integrates with the BD-J UI framework. The text is rendered using a vector-based font either coming from the disc, the player (default font) or downloaded from the network.
- Button sounds from HDMV can also be used by the Java UI framework. Sound files can be loaded and rendered as a reaction to the user pressing a key, or as a reaction on a marked event related to the film—or as a reaction to any event generated by a BD-J Application.
- Authenticated applications can use a (signed) permission request file to acquire permissions that go beyond the BD-J sandbox. Permissions can be acquired for:
  - Reading and writing to local and system storage
  - Using the network connection (to connect to defined servers)
  - Access of the file system on the BD-ROM disc
  - Title selection of other titles on the BD-ROM disc
  - Control of other running BD-J applications
- BD-J applications can use the `java.net` package to connect to servers on the Internet. The physical connection might differ between implementations e.g. Ethernet, telephone line, etc. At the network level, TCP/IP is supported and the HTTP protocol may be used. Moreover, the Java package for secure connections is included (JSSE) as part of the BD-J platform. Before a BD-J application can use the network connection, it must be authenticated and have suitable permission to use the network.
- The websites to which the application will go are under full control of the Content Provider. This control is guaranteed in two ways:
  - Only (disc) authenticated BD-J applications are allowed to run when the disc is played. The application controls the use of the network connection.
  - In addition, permissions defined on the disc can restrict the use of the (TCP/IP) network connection to certain sites.
- BD-J will include support for storage. Two flavors of storage are included: mandatory System Storage and optional Local Storage. All storage is accessed using methods from the Java IO package. The path for local storage is as specified by [GEM].
- System storage is storage that will be present in all BD-J players. The required minimum size of this system storage will permit storage of application data like settings, high-scores etc. It will not be big enough to store downloaded AV material. For this purpose, optional local storage is available. Typically system storage will be implemented using Flash memory and the optional local storage will be implemented on a HDD.
- Since storage is a shared resource between all discs played on the player, Java access control is part of BD-J. BD-J applications can only access a disc specific part of the storage space and cannot access the part belonging to other discs.

Content development

Content authors have a variety of development strategies available, including the use of traditional Integrated Development Environments (IDEs) like NetBeans or Eclipse, non-programming graphical environments similar to Macromedia Director, or via rendering engines which consume standard data formats such as HTML, XML, or SVG. Having a full programming environment available on every Blu-ray Disc player provides developers with a platform for creating content types not bound by the restrictions of standard DVD. In addition to the standard BD-J APIs, developers may make use of existing Java libraries and application frameworks, assuming they do not use features outside the constraints of the BD-J platform, include that Java ME only supports Java version 1.3 class files.

A set of freely available tools that allow Java developers to produce complete disc images incorporating BD-J is available from the HD Cookbook Project.[3] In order to test content in a typical development environment (MS Windows), one needs either a PlayStation 3 or a third-party software player for Windows, paying attention to player versions to ensure that the player supports BD-J.[4][5][6]

Because of the many different standards and components involved, creating unified documentation on BD-J has proven to be a challenge.[7][8][9]

Sample code

The BD-J environment is designed to run Xlets with non-`javax.*` packages available to take advantage of the features particular to this platform beyond that defined by Java TV.

Even a simple example such as FirstBDJApp.[10]

A developer might choose to use not `javax.*` packages and instead use:

1. HAVi classes in package tree `org.havi.*`: alternative classes to obtain, for example, an `org.havi.ui.HScene` far beyond what is provided by `javax.tv.graphics.TVContainer`(they are both extensions of `java.awt.Container`)
2. Digital Video Broadcasting (DVB) classes in package tree `org.dvb.*`: alternative classes to, for example, the `org.dvb.event.UserEventListener` interface rather than `java.awt.event.KeyListener` for support for key presses and keycodes specific to popular CDC devices.
3. Blu-ray Disc classes in the package tree `org.bluray.*`: the DAVIC and DVB classes depend upon to recognize additional events peculiar to the BD-J platform such as popup menus and to locate media on the Blu-ray disc.
4. DAVIC API classes in package tree `org.davic.*`: A small set of classes wrapping or extending other network and media resources peculiar to interactive TV the HAVi, DVB and Blu-ray classes use for locators and specialized exceptions beyond the realm of JMF (such as content authorization).

A working example of a program using some features from each of the class trees would be the BdjGunBunny Xlet (a very simple version of Space Invaders using an image of a rabbit as the shooter and turtles as the targets) provided as an example in the Java ME 3.0 SDK.

```
import javax.tv.xlet.XletContext;

import org.havi.ui.HScene;
import org.havi.ui.HSceneFactory;
```

```
import java.awt.Container;
import javax.tv.graphics.TVContainer;

// Getting a container for the screen could be

public void initXlet(XletContext context) {

// Java TV API to be compatible with Java TV
TVContainer scene = TVContainer.getRootContainer(context);

// Or for BD-J, to utilize HAVi features not available in Java TV
HScene scene = HSceneFactory.getInstance().getDefaultHScene();

// Or perhaps more generally...
Container container = null;
boolean realBDJ = true;
if (realBDJ)
    container = HSceneFactory.getInstance().getDefaultHScene();
else
    container = TVContainer.getRootContainer(context);
...
}
```

and the same for the other non-`javax.*` packages. Likewise, when trying to play a video, one might call the Blu-ray and DAVIC utility rather than using generic JMF:

```
import javax.media.Player;
import org.bluray.net.BDLocator;
import org.davic.media.MediaLocator;

   MediaLocator stars = new MediaLocator(new BDLocator("bd://0.PLAYLIST:00003"));
   Player player = Manager.createPlayer(stars);

// Rather than traditional and portable but more limited pure JMF

import java.net.URL;
import javax.media.Manager;
import javax.media.Player;

    Player mediaPlayer = Manager.createRealizedPlayer( new URL("file:/mymovie.mov" ));
```

Related publication

- *Programming HD DVD and Blu-ray Disc The HD Cookbook* (2008) by Michael Zink, Philip C. Starner, Bill Foote - ISBN 978-0-07-149670-4 - book website

See also

- Advanced Content, BD-J's counterpart on HD DVD
- Blu-ray Disc

References

1. "Blu-ray BD-J Application Development using Java ME". Archived from the originalon 2008-09-11. Retrieved 2008-10-21.
2. "Blu-ray Disc Assn. promotes new Bonus View".
3. HDCookbook for BD-J

4. "Blu-ray Disc Java and GEM/MHP/OCAP Authoring Notes and Guidelines". Archived from the original on 2008-12-24. Retrieved 2008-10-21.
5. Getting (BD-J) PC Player softwareArchived 2008-12-24 at the Wayback Machine - mentions that special "developer versions" of the play might be required
6. HelloWorld BD-J Application: Your first cup for the next generation DVD Archived 2009-02-07 at the Wayback Machine - includes notes on enabling remote debugging or logging with PC players
7. Unified Developer Documentation for BD-J now possible
8. Building Javadoc API Reference Documentation for Blu-ray Disc Application Development
9. "Getting a BD-J Platform Definition". Archived from the original on 2008-12-24. Retrieved 2008-10-21.
10. Blu-ray Disc Application Development with Java ME, Part 1: Creating Your First Application Bruce Hopkins, September 2008

External links

- Official java.net BD-J Forums - Official Sun java.net Forums for Blu-ray Disc Java.
- bdjforum.com - Unofficial forum for BD-J developers and issues surround HD authoring.
- JavaOne 2007 Technical Sessions: Producing Blu-ray Java Software Titles for Hollywood
- Official website for DVB-MHP and DVB-GEM - Open Middleware for Interactive TV
- TV Without Borders - MHP/OCAP Website from Steven Morris.
- HD Cookbook - Code and other recipes for Blu-ray Java, GEM, MHP and OCAP
- Alticast BD-J Solutions