Desktop Apps: A Zero Deployment Strategy

By Brian Hipple *BASIS Advantage 2023*

This latest BASIS Advantage article is the last article in a series that started with Classpaths and was followed by the new Classloader. Desktop Apps, first introduced in BBj® 19.20, is the BASIS replacement for Java Web Start (JWS). JWS is a framework developed by Oracle that allows users to start Java applications directly from the internet using a web browser. Oracle deprecated JWS in Java 9, and starting with Java 11, they removed JWS from their distributions. Since BASIS builds BBj on the Java platform and many BBj deployments made use of JWS, we needed to find an alternative and perhaps even an enhanced solution for the BBj community. In this article, we will provide an overview of Desktop Apps, including why they are needed, what they are, the benefits they provide, how they work, how they've matured in later revisions, including the latest BBj 23, and how they fit into BASIS's Zero Deployment strategy.

The Trouble with JWS

Let's be honest: JWS was never a perfect solution. System administrators had to create and manage Java Network Launch Protocol (JNLP) files on the server, and these application definition files could get quite complicated with their many different settings—most of which were never needed or used by BBj applications. JWS also required the generation and inclusion of a Web

Start security certificate to run the application, and the JAR files had to be signed before the client could download them.

The JnlpExePacker

BASIS provided an interim solution with the JnlpExePacker, a BBj application that creates native Windows/macOS/Linux applications given a JNLP file, it includes the necessary BBj JARs and a JRE. When users run the resultant native application with BBj 19.10, it takes advantage of the new BBj classloader (see the "Loading with Class" article) to dynamically download needed classes/files to the client. Any JNLP/BBj/JRE update requires regeneration and redeployment of the native applications. However, the JnlpExePacker solution requires JNLP files and, of course, those have all of the limitations of JWS that we previously mentioned. Therefore, we decided to work smarter, not harder, and the idea of Desktop Apps was born.

Desktop Apps: The Smarter Solution

When designing Desktop Apps, we wanted to include the concepts that we liked about the JnlpExePacker and JWS, remove the limitations, and add new update capabilities. The functionality we incorporated was the definition/management of applications on the server via the Enterprise Manager (EM), the initial installation and running of the program via a URL, and the utilization of the new BASIS classloader to download needed resources from the server. We removed the limitations of having complicated JNLP files, the requirement of a security certificate, and the JAR signing. We also added the ability for updates to Java, BBj, or programs to be automatically handled by the client, and for all configured applications to be available from the BBj Jetty home page.

Creating a Desktop App

So, how do you create a Desktop Application? You can do it in two simple steps:

Step 1: In the EM, specify a JRE for each client-side Operating System; Windows, macOS, and Linux that your clients use. You only need to do this once, no matter how many Desktop Apps you will create. Of course, you will want to update these JREs as new versions become available and supported, and the Desktop App clients will be automatically updated! **Figure 1** displays the configured JREs in Enterprise Manager.

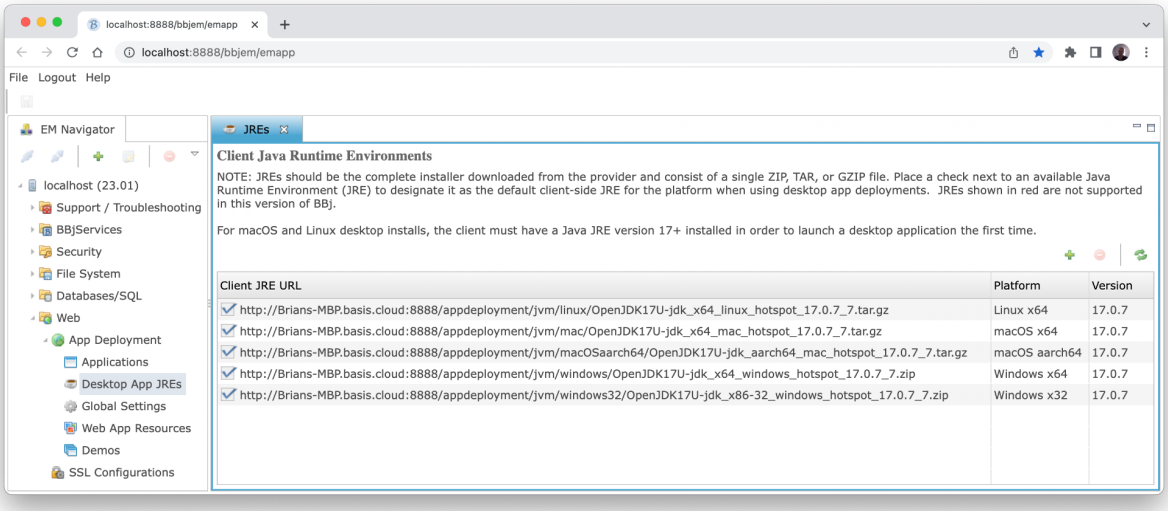

Figure 1. *Defining JREs for a Desktop Application in Enterprise Manager*

Step 2: Configure the BBj application for the Desktop App to run, including a unique name, program file, program arguments, working directory, configuration file, load image (splash-screen) and shortcut images, and BBj settings. This should be familiar for system administrators as this is almost the same process that they use to configure a BUI (Web) App. **Figure 2** shows the Desktop App definition for Addon.

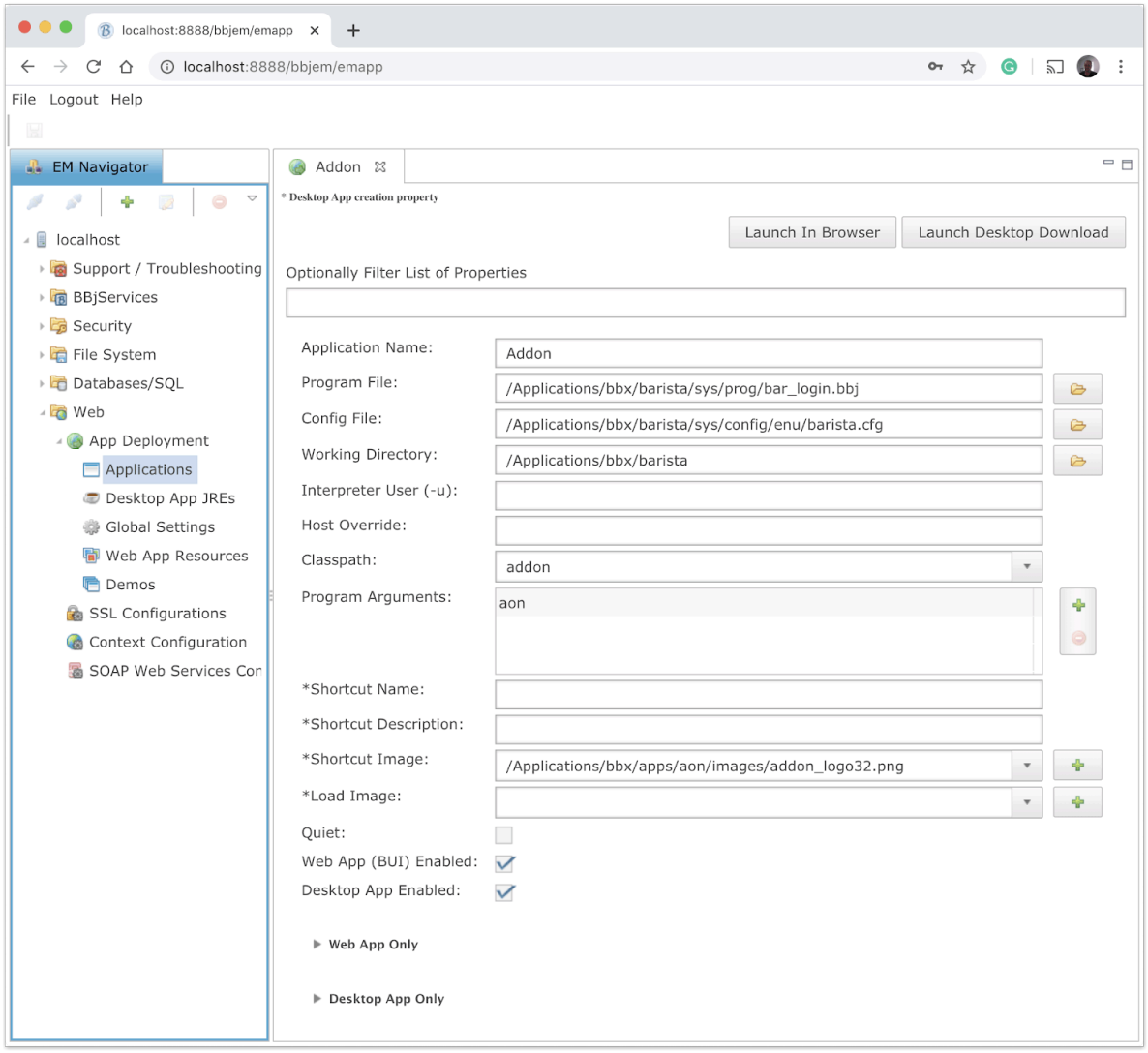

Figure 2. *A Sample Desktop Application in Enterprise Manager*

After creating and enabling the Desktop Application, the URL to run the Desktop App is available from the default Jetty home page, shown in **Figure 3**, and from the EM's Applications list.

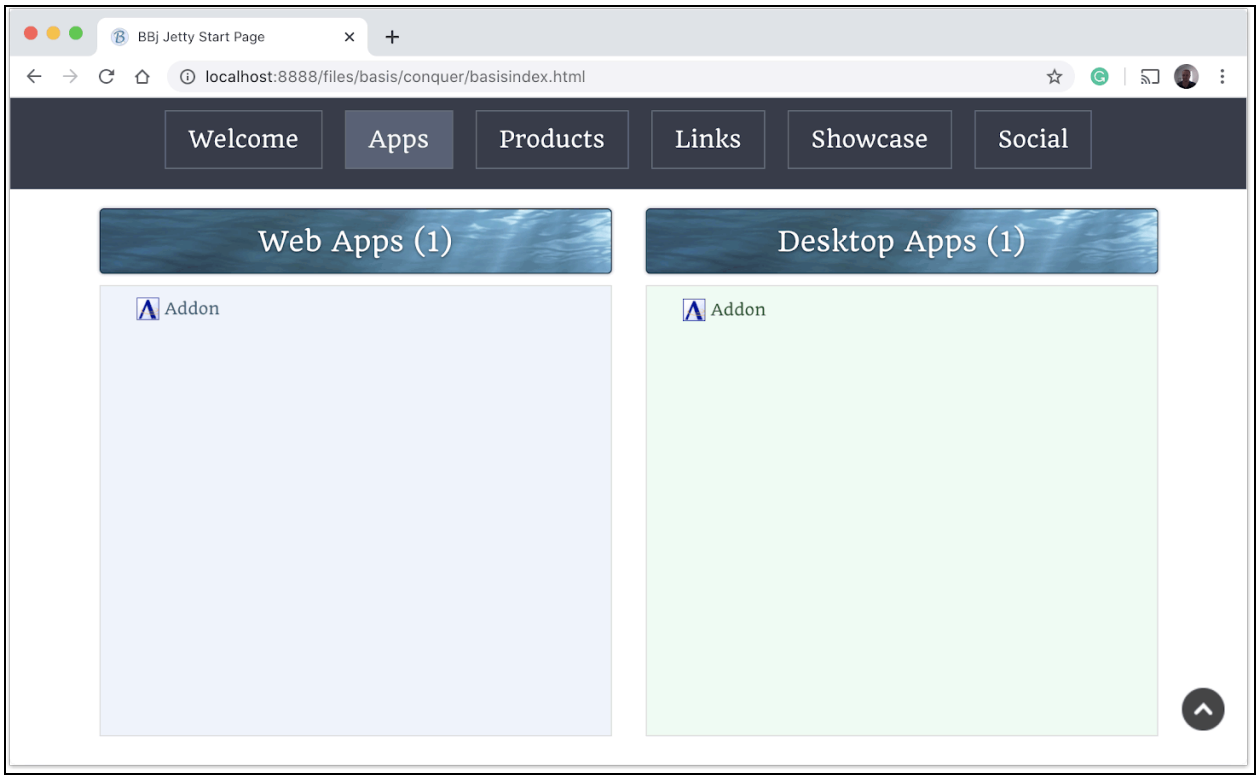

Figure 3. *Defining the Application in Enterprise Manager*

When a user clicks on the application link displayed on the Jetty home page or opens the associated URL in a browser, the Desktop App will be downloaded, installed, and run on the client machine. During the installation process, it creates an application on the client along with a desktop shortcut to subsequently run the Desktop App. Each time a user runs the Desktop App, it automatically checks for and obtains BBj, JRE, and program configuration updates from the server. **Figure 4** shows the Addon Desktop app launching as a native application from the user's desktop shortcut.

Figure 4. *AddonSoftware's Splash Screen*

New Functionality with BBj 23

In BBj® 23.00 Desktop App, JRE configuration enhancements were made in EM, where multiple JREs can now be added at once, only JREs on which BBj is supported will be allowed, and the version of the JRE and platform will be automatically determined and displayed. JREs previously added, that the current version of BBj isn't supported on, will be displayed in red. Desktop App client JRE validation has also been added, where clients now perform JRE validation, and if the version of BBj is not supported on the JRE, a message will be displayed to the user. There are also more EM configuration options to control the display load image and text, as well as whether to show the run message after the initial installation. Remote cache cleanup is also now available. When a remote client is run (usually as a Desktop App), a Java class cache is maintained (per BBj version) on the client. This new EM option Web>App Deployment>Global Settings allows for the removal of older version cache(s) on the client when a Desktop App is run. **Figure 5** shows

the Remote Cache Cleanup option.

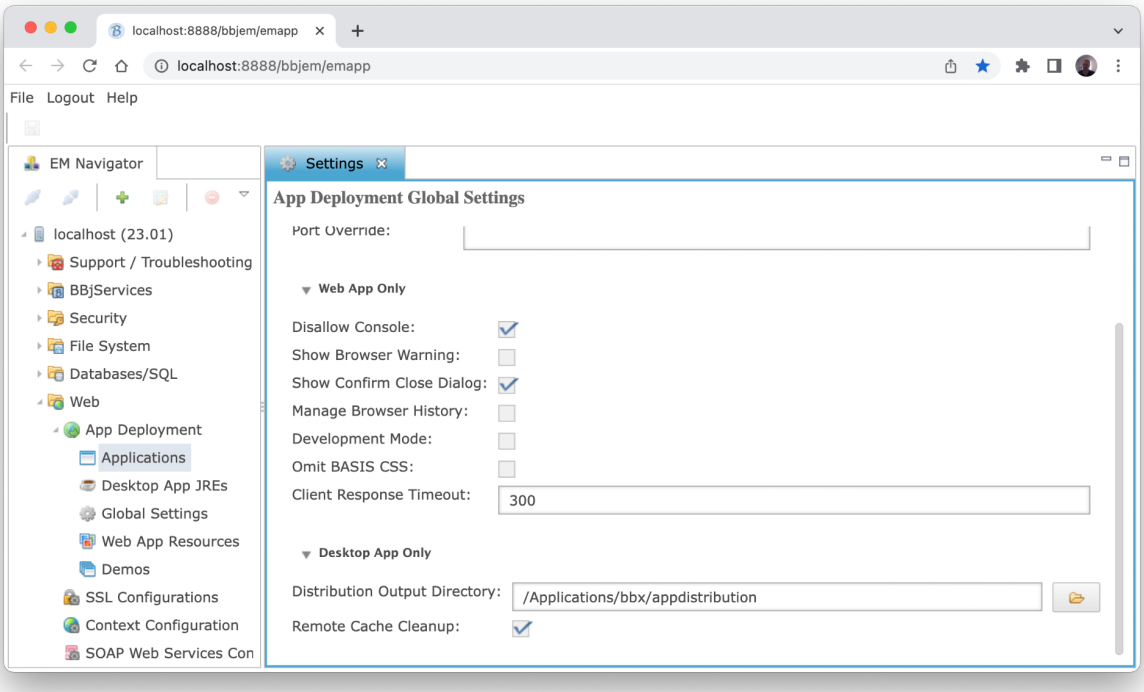

Figure 5. *Distribution Output Directory and Remote Cache Cleanup Options*

The most significant new functionality is the ability to create a Desktop App distribution, where much like the aforementioned JNLPExePacker, it will create a directory structure (and compressed archive files, either a zip for Windows or tar.gz for non-Windows) for each type of JRE configured that can be used by a client to run the Desktop Application. This option provides an administrator the ability to deploy these directories/archive files to client machines in the manner they choose, controlling such things as where they would like them to be installed, shortcut creation, and customizations. The distribution is created under the directory specified in the global options (see **Figure 5**) when the Build Distribution button is clicked while editing a Desktop App in EM. Desktop App clients deployed this way will still be updated when run, so any updates to JREs, BBj, or configuration will be applied automatically. **Figure 6** shows the Build Distribution button and the message when the distribution is successfully created.

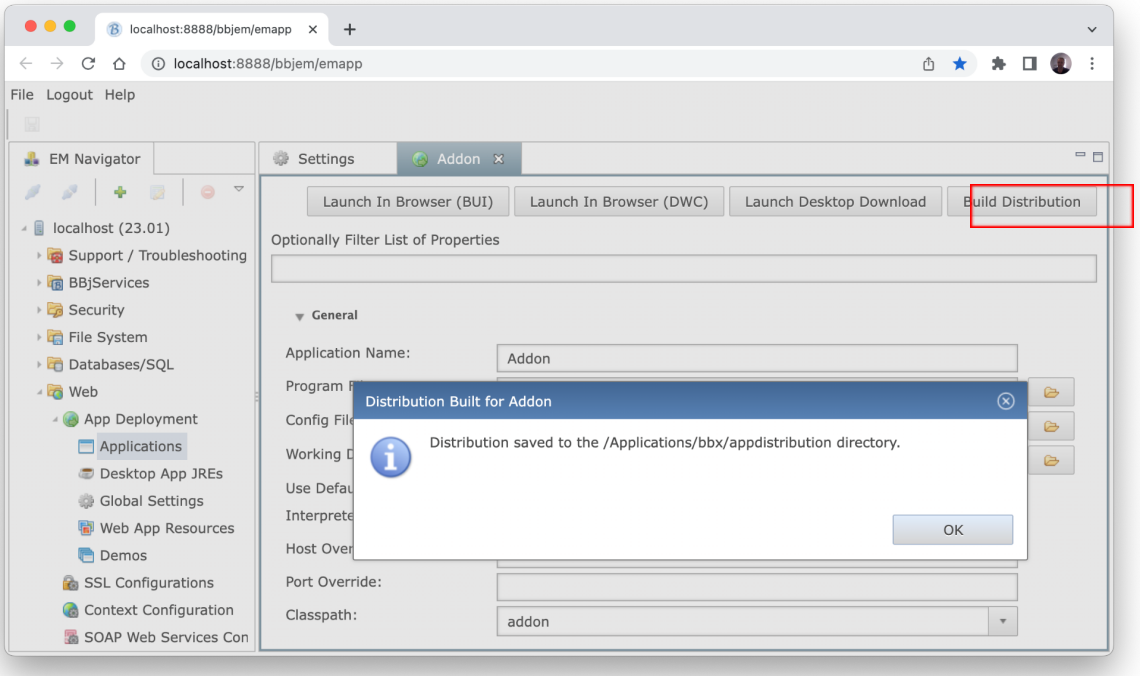

Figure 6. *Build Distribution Button Message*

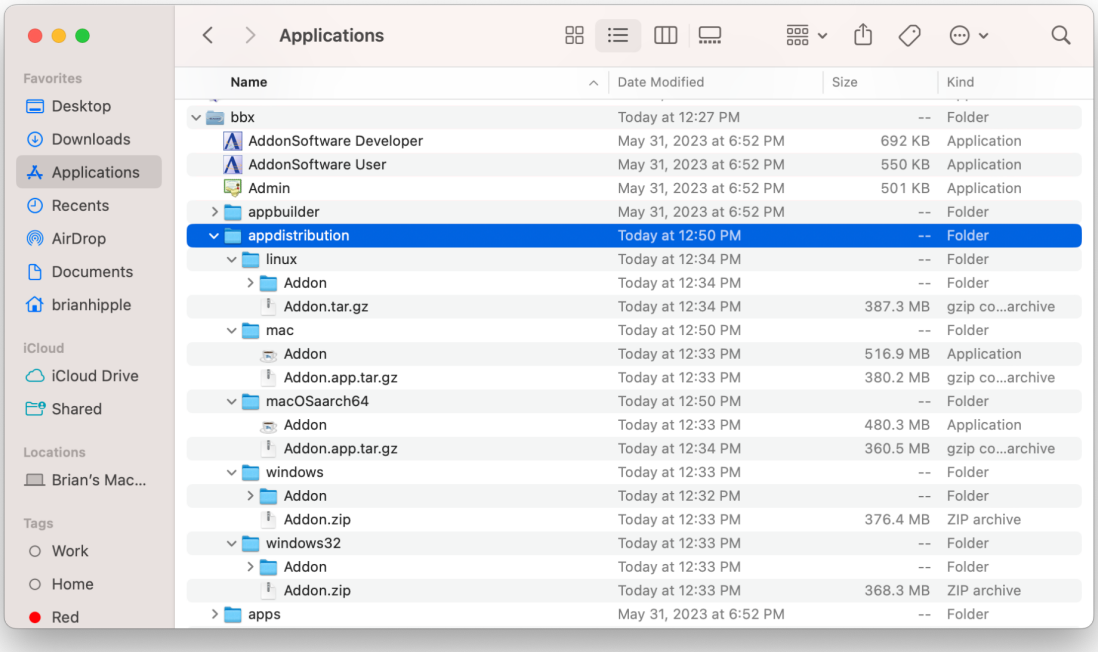

Figure 7. *Directory and Compressed Archive Files for Each Client Type*

Summary

The BUI (Web) client BASIS already delivers on the promise of Zero Deployment, where the user only needs a browser and doesn't need Java or BBj installed to run the application.

Do Desktop Apps now also deliver a true Zero Deployment solution for running a BBj thin client on the desktop? For Windows, this is absolutely the case. BBj creates the Desktop App as a native Windows executable on the server and downloads it to the client. No Java is necessary for the client for initial installation. For macOS and Linux clients, BBj creates an executable JAR file and sends it to the client thus, a supported version of Java is still required on the client for initial installation. Therefore we can say that it is a Zero Deployment solution for Windows clients and subsequent upgrades for other clients and a “Near”-Zero Initial Deployment solution for non-Windows platforms.

Desktop Apps provide a straightforward and greatly enhanced deployment methodology for delivering BBj applications to the desktop that overcomes all of the disadvantages of Java Web Start.