action™ engine

Action Grid Skin Development and Customization Guide

Action Grid Skin Development and Customization Guide

8520 154th Avenue NE
Redmond, WA 98052
+1-425-498-1500

Contents

1. Getting Started

Introduction to Action Grid

Action Grid is software that provides mobile operators with an opportunity to deliver a branded User Experience Layer on smart phones. This flexible solution helps to elevate brand, control the customer's service experience, and generate additional data revenue. Action Grid drives data service usage by not only bringing users closer to the content they care about, but also by delivering that content via over-the-air (OTA) mechanisms.

Overview of an Action Grid Skin

Action Grid enables mobile operators to easily brand their offerings with *skins* that simulate a smart phone shell, or user interface. These skins are highly customizable, yet they provide a consistent user experience across various devices and operating systems.

While skins can appear vastly different, they all perform similar tasks: launching applications, displaying status indicators, and providing optimized navigation.

Skin Elements

A skin defines numerous elements of the user interface, including screen background, layout, icons, labels, menu hierarchies, selected state behaviors, font styles, and colors. Optionally, a skin can also define sounds to play when events occur.

UI Customization Levels

Action Grid supports several levels of user interface customization in order to meet the needs of both the mobile operator and the end user. The levels of customization include:

- Skins using standard grid and list view layouts with icons that have rollover or selected state behaviors.
- Skins with an asymmetrical custom layout and animated icons.
- Unique skins that can define new UI approaches and designs that need not follow any traditional paradigms.
- Skins with multimedia elements such as sounds.

Phones Supported

Action Grid will run on devices that use the following software platforms:

- Microsoft Windows Mobile™ 2003 for Pocket PC Phone Edition
- Microsoft Windows Mobile™ 2003 for Smartphone
- Microsoft Windows Mobile™ 2002 for Pocket PC Phone Edition
- Microsoft Windows Mobile™ 2002 for Smartphone
- Symbian OS v6.1/Series 60 v1.2, Nokia Edition (Nokia 3650)
- Symbian OS v7.0s/Series 60 v2.0, Nokia Edition (Nokia 6600).

Audience

This guide is written for software developers, graphic artists, and designers who will be creating or modifying Action Grid skins.

Required Skills

To create or modify Action Grid skins, software developers should understand HTML, cascading style sheets (CSS), XML, and JavaScript.

Graphic artists and designers should understand and have access to image editing software that can work with bitmapped images in GIF, bitmap (.BMP), or JPEG format files. These tools include industry standard software such as Adobe PhotoShop, Macromedia Fireworks, and Jasc Paint Shop Pro. Knowledge of image transparency issues and graphics file optimization techniques is needed to achieve the best results.

Required Tools

The minimum required development tools include:

- Microsoft Windows 2000 or later.
- A text editor for editing HTML, CSS, XML, and JavaScript documents.
- An image editor for editing graphics files, such as bitmaps and GIF images.
- A tool for creating compressed files in standard Zip format.
- A tool for creating a Globally Unique Identifier (GUID).
- Devices for development and validation of the software.

Setting Up a Development Environment

The developer toolkit includes the sample skins, the Windows desktop client `Grid.exe`, and the Action Grid clients for the supported devices.

Typographic Conventions

This guide uses the following typographic conventions:

Convention	Explanation	Example
Arial Bold	Indicates Graphical User Interface elements.	Press **Go** to load the **User Details** window.
`Courier New`	Indicates code samples, screen display, or network and file paths.	`//Admin/Device Support/ Users/Address`
`Courier New Bold`	Indicates user input, contrasted with on-screen output.	Enter User Name: **johndoe**
Times New Roman Italics	Indicates book, section, and software titles.	Refer to the *Create New User* section.
<*`ItalicType`*>	Indicates variables to be replaced with the user's values.	Enter <*`DeviceName`*>
>	Indicates menu and tree hierarchy.	To open a new document, go to File>Document>Open

NOTE: Notes provide additional information or references.

2. Anatomy of a Skin

Action Grid Architecture

The architectural design of Action Grid is split into two main components: Action Grid client software and Action Grid server software. The Action Grid client software is an application installed on the user's device to display and manage the interaction with the user interface. The client software has the primary responsibility of providing convenient access to device capabilities and data services via an XML and JavaScript-driven interface. It renders the skin pages using resource and layout information defined in the Action Grid skin markup language, *Skin XML*. Finally, it performs the scripted actions when various events occur, similar to how a Web browser may navigate to and render a Web page when a hyperlink is clicked.

There are distinct versions of the client software for the different supported device platforms. In addition to the client software for the devices, there is a version of the client software that runs on a standard PC running Microsoft Windows 2000 or later. This Windows desktop client serves as an emulator for Pocket PC and Smartphone devices.

The Action Grid server software is documented in *Action Engine Administration Guide*.

Action Grid Skin Files

Directory Structure

During Action Grid skin development, the files in a skin are placed in a specified directory structure on the skin developer's computer. Directory names in regular font are standard names determined by Action Grid, and directory names in italic are names chosen by the skin developer.

```
C:\Action Engine\Grid
   __grid
         PocketPC
         Series60
         Smartphone
   previews
         SkinPreview
   skins
         SkinName
               PocketPC
               Series60
               Smartphone
```

The `__grid` directory contains files used by all skins. These files provide standard functionality including user interface strings, as well as displaying a skin chooser, a message box, a progress indicator, and a file chooser. The files in this directory should not be modified although your custom skin can override their behavior.

The `previews` directory contains skin preview images for display by the Action Grid server software.

Action Grid skins are in directories under the `skins` directory. The skin developer chooses the directory name for each custom skin. Like `__grid`, the skin files are typically placed in the directories corresponding to the supported platforms, such as *`PocketPC`*, *`Series60`*, or *`Smartphone`*. The skin developer chooses these directory names; the names will not appear on the device or anywhere outside the development process. All of the files that comprise the skin – HTML, CSS, XML, and image files – are stored in one folder. This makes each skin its own self-contained package with a complete set of all necessary resources. The compartmentalized nature of this architecture aids in the creation, revision, organization, and publishing of each skin.

XML Files

Each skin contains a central file named `skin.xml` that defines the skin's name, home page, background image, and language settings. To ease localization, a skin may also have an XML data file containing user-visible strings for the supported languages.

Artwork

Artwork must be provided for each skin and platform combination. The artwork will typically include

- Background images
- Icon images in selected and unselected states
- Battery strength indicator
- Signal strength indicator
- Other status indicators associated with various device functions.

HTML Files

HTML files define the skin home page and menus. These files specify the images to display as the background and icons. Finally, these files contain JavaScript to handle events such as key presses and focus changes.

Cascading Style Sheets

Like standard style sheets, an Action Grid Cascading Style Sheet (CSS) contains a set of style rules that specify how to display various elements in a document. Using a style sheet facilitates quick and easy manipulation of many display characteristics in the presentation layer. An Action Grid CSS is an XML file with a root element named `<css>`. HTML documents in a skin reference CSS files to apply the style rules. The styles can also contain JavaScript for handling events.

Grid.xml

A file named `grid.xml` resides on the device. It contains the skin name, background, and other settings. It is located at the `C:\Action Engine\Grid` level in the directory structure. This file will rarely be edited directly by the skin developer.

Sample Action Grid Skins

Action Grid includes several sample skins that demonstrate various features and layout options. Most of the skins are provided for Pocket PC, Symbian Series 60, and Smartphone. The sample skin files are in directories under `C:\Action Engine\Grid\skins`.

The sample skins are provided as examples of design approaches, as well as reusable templates to jump-start the creation of new skins.

Skin Elements

Art Files

Art files for skins may be bitmaps with the extension .bmp, GIF images with the extension .gif, or JPEG images with the extension .jpg. The choice between these file formats depends on several factors, the most prominent being the tradeoff between image quality and file size. File size is especially important when skins will be deployed over-the-air (OTA).

Background

The current Action Grid skin determines the default background. This image can be thought of as the "stage" that determines the overall thematic look and feel. It also provides the frame against which all the interactive elements such as icons and *fly-out menus* are designed and used. Figure 2-1 shows the background images from the YourBrand, Candy, and Rugby skins for Pocket PC.

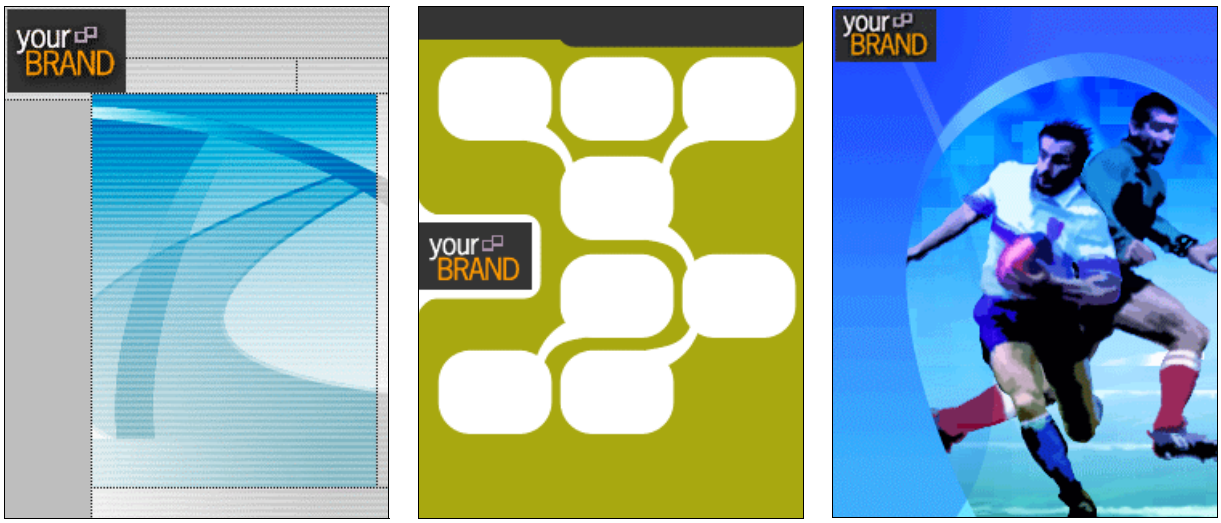


Figure 2-1 Example background images for Pocket PC

Icons

The user clicks an icon in the Action Grid skin to display a menu, launch an application, or open a file. The icon images can be any size. A caption should be supplied for each icon, and the caption can be either hidden or displayed above, below, or on either side of the icon. Figure 2-2 illustrates two sample icon images in both unselected and selected states.

phone_btn.gif phone_btn_over.gif contacts_btn.gif contacts_btn_over.gif

Figure 2-2 Example buttons in unselected and selected states

Fly-Out Menu Backgrounds

When an icon is selected, the skin may display a *fly-out menu* populated with hyperlinked text in list format. Figure 2-3 illustrates some sample fly-out menu background images.

Figure 2-3 Example fly-out menu background images

Soft Key Labels

A *soft key* is a key on the device assigned to software shortcuts. For some devices, the user manuals refer to these keys as *selection keys*.

In the Action Grid sample skins, pressing the right soft key selects an “Options” menu. Figure 2-4 illustrates some sample Options button images from the YourBrand and English Football Club skins. The buttons are shown in both unselected and selected states.

options_btn.gif options_btn_over.gif options_btn.gif options_btn_over.gif

Figure 2-4 Example Options button images

Text

Text can appear in the skin in several places, including icon captions, menus, message boxes, and others. Text can be either static or hyperlinked. Users can navigate to and select hyperlinked text, such as menu items, to invoke an action. Action Grid provides localization mechanisms to enable the developer to segregate text strings by language.

Predefined Status Indicators

Action Grid provides predefined status indicators for battery status, signal strength, appointments, and several others. These are linked to current device states and give an accurate graphical representation of status to the user. Some status indicators, such as signal strength, are always visible. Other status indicators only display when active. An example is the missed call indicator that appears after an incoming call was not picked up.

Putting It All Together

Figure 2-5 shows several of the YourBrand images: the skin background, several icons, a fly-out menu, and a soft key Options button icon.

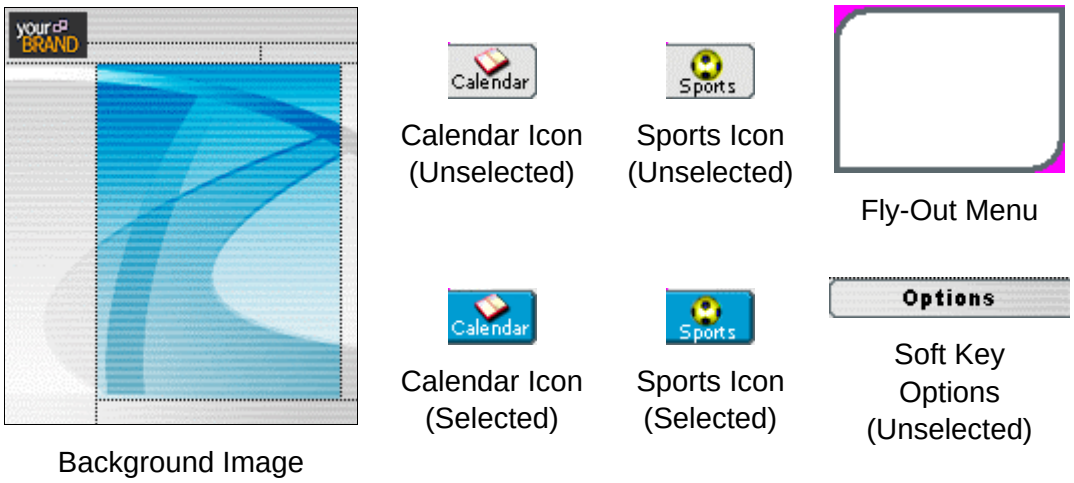


Figure 2-5 Images from the YourBrand skin

Figure 2-6 illustrates the completed skin with icons, status indicators, a fly-out menu, and a soft key Options button icon.

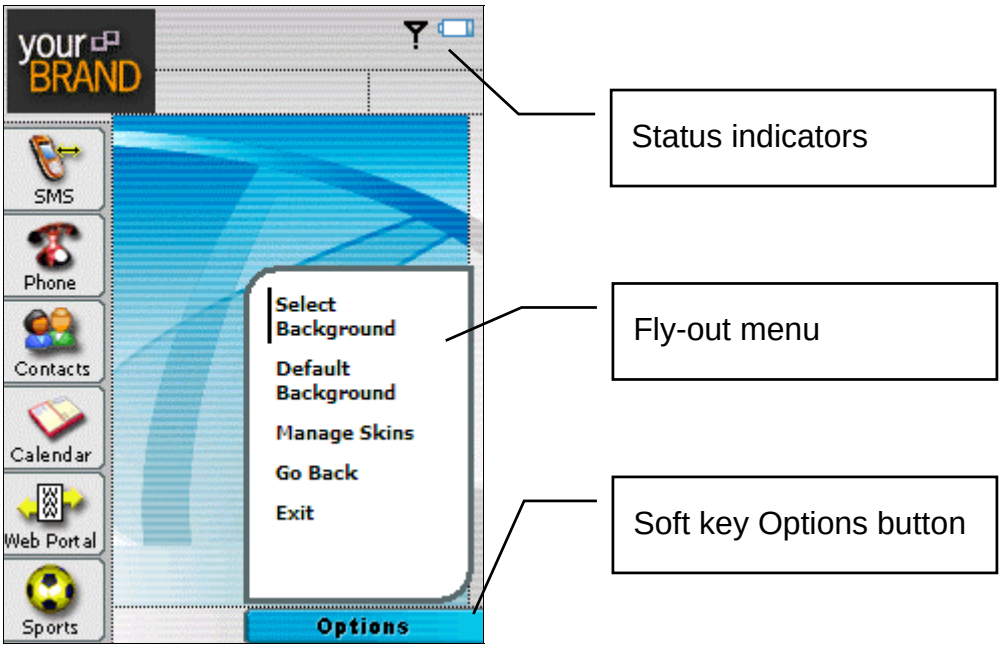


Figure 2-6 The completed skin

Sounds

In addition to providing images and navigation, an Action Grid skin can optionally provide sounds associated with the skin's theme. The set of supported sound file formats depends on the target platform. At a minimum, all platforms support .wav files. The sounds can be associated with events, such as focus changes, to enhance the overall experience. An example might be a short burst of applause or a sports announcer's voice when a Sports icon is selected. Care should be taken not to abuse this for both aesthetic and file size reasons.

3. Skin Development Workflow

Workflow Description

The basic steps in developing and validating a custom Action Grid skin are:

- **Planning:** In this initial step, numerous questions must be answered. What are the customer and end user requirements? On which platforms will the skin be deployed? What functionality will be provided? Will the skin have custom sounds? What is the development schedule? What are the legal or regulatory requirements for desired image use? Thorough planning to decide the answers to these questions and others aids the subsequent workflow.
- **Design:** Design the visual and functional aspects of the user interface. Sketch out the artwork for background, icons, and other images. Decide the exact pixel sizes and layouts of the images. Determine the menu items and all navigation behavior. At this

stage, it is often beneficial to create static screen mockups of the design to ascertain its suitability or desirability as well as to shorten the actual production process.

- **Implementation, Packaging, and Validation:** Create the artwork and write the HTML, style sheets, and XML files. The HTML files contain the information to associate the user interface and images with navigation, sounds, and other actions. After the implementation is complete, the skin must be packaged for validation.
- **Deployment:** The Action Grid client software and skins may be deployed to a device several ways. In addition to installing the skin on customers' devices, skins must be installed and administered on the mobile carrier's network. After the skin is on the device, it must be validated for correct display and behavior.
- **Support and Maintenance:** After deployment, mobile carriers must plan for ongoing support and maintenance.

Skin Planning

Customer and End User Requirements

There are many questions that must be answered for requirements planning. These questions may include, but are not limited to

- What is the target date for skin deployment?
- What devices will be supported?
- What are the functionality requirements for users?
- Who is the target audience and how will the skin address their needs or desires?
- What are the branding and logo requirements?
- Is there existing artwork that can be used, or must it be created?
- What is the general layout of the user interface?
- What languages must be supported? Will the artwork need to be localized?
- What additional media is needed? Will there be custom sounds?
- How will the skins be deployed?

Single Device or Multiple Devices

If the Action Grid skins are going to be deployed on multiple devices, prior planning is important. Because the screen sizes in the currently supported devices have different sizes and aspect ratios, designs and images typically cannot simply be scaled up or down to fit.

In most cases it is better to scale images down rather than up, so it may be best to start with the device platform featuring the largest screen, Pocket PC, and work from there. Sketching screen mockups that include navigational elements will help give a sense of how the skin will look on the device, as well as provide some of the raw materials for production of the skin.

Although there are significant differences between the currently supported device platforms, there are also many similarities which can aid in the design and workflow of Action Grid skins. For example, there are only small differences in the size and aspect ratios of the Symbian and Smartphone screen sizes. With forethought, one design can yield images to be used for both platforms. In this respect, Smartphone and Symbian have more in common with each other than with the larger Pocket PC platform. This implies that by working concurrently with these two apparently unrelated devices, significant time savings may be realized.

Phone Platform Considerations

Windows Mobile

- Smartphone, 2002 and 2003
- Pocket PC Phone Edition, 2002 and 2003

Microsoft offers two mobile device platforms: Pocket PC Phone Edition and Smartphone. The major differences between the two can be explained by their primary intended functionality. The Smartphone is first and foremost a phone device for making and receiving voice communications. The Pocket PC Phone Edition was designed as a small personal computer with additional phone capabilities. For developers creating Action Grid skins, the most significant challenges are likely to involve the differences in their respective screen sizes and aspect ratios.

Both Pocket PC and Smartphone support GIF, JPEG and BMP image formats, so these are good choices for building Action Grid skins.

Symbian

- Symbian OS v6.1/Series 60 v1.2
- Symbian OS v7.0s/Series 60 v2.0

Symbian is the primary operating system platform for all Nokia devices. The kind of device and its release date will generally impact which version of the Symbian platform it runs. Action Grid supports two versions of the Symbian OS: v6.1/Series 60 v1.2, and v7.0s/Series 60 v2.0.

Because Symbian devices have the smallest screen, it is important to design considering this constraint, and then create mockups for it using the design of the Pocket PC as a template. Elements from this can be cut and scaled to assemble the basic Symbian mockup. If care is taken at this point, many of the image assets will be nearly ready for use in production.

All devices running these systems have a screen size of 176 pixels wide by 208 pixels tall. Compare this to the Smartphone screen at 176x220 pixels and it is easy to see how parallel development could be beneficial to design and workflow.

Skin Design

Once the planning is complete, detailed skin design can begin.

- The designer typically creates several designs or mockups for review. These should be done at the actual size and resolution of the intended device platforms.
- For each supported device, the design should include

  - Exact position and size of each icon and button
  - The selection action and navigation behavior of each icon and button
  - Exact position and size of the fly-out menus
  - Transparency of icon, button, and menu images
  - The contents of each menu
  - The selection action and navigation behavior of each menu item.
- This phase is often iterative, with changes being requested until the final design is approved.

Skin Implementation, Packaging, and Validation

Completing the design phase carefully and thoughtfully will ease the implementation and validation phase of Action Grid skin development.

Creating the Artwork

At the most basic level, skins contain separate image files for the background, navigation icons, and fly-out menus. Depending on the design, there will be additional visual assets, such as status indicators. These individual images are then reassembled at run time to display the skin.

- Using an image editing program, such as PhotoShop, the graphic artist or designer slices the screen design into the component parts used in the skin and optimizes them to reduce file size. This will include the larger background image as well as smaller ones that define the icons in both the selected and unselected states. There may also be images for the fly-out menus and status indicators. Most images will be saved as GIF format files, but this is not a requirement.
- If a sample skin supplied by Action Engine is being used as a template, replace the images in the skin folder with the newly created ones. The easiest method is to use the same file names as the sample skin so fewer changes are required in code.
- When the first skin is nearly finished, the designer or developer can run the desktop Action Grid client by starting `Grid.exe` located in the `C:\Action Engine\Grid` directory. This allows the designer to preview and adjust the skin as it evolves. It is also one of the first steps in the testing and validation process.

Sounds

Sounds can be added as desired and are triggered by events using JavaScript in much the same way as that used to make icons change states on focus or selection events. Judicious use of sound can enhance the user experience without adding unduly to the overall skin size, but care should be taken not to overdo it. Production of sound files for an Action Grid skin is subject to the standard size and file type constraints of the individual mobile devices.

CSS Files

An Action Grid Cascading Style Sheet is an XML file with a root element named `<css>`. HTML documents in a skin reference CSS files to apply the style rules. In the sample Action Grid skins, the CSS files for one skin typically total roughly 100 lines defining about 15 styles.

HTML Files

One HTML document is required for the skin's main "home" screen and one for each fly-out menu to be displayed. The HTML documents define several aspects of the skin, including the position of images, the text of menu items and captions, and the navigation behavior on key presses. The CSS, HTML, and images are closely integrated, so they are usually developed simultaneously. In the sample skins, there are 5-7 HTML files containing 300-500 lines per skin.

XML Files

Each skin must contain a file named `skin.xml` that defines the skin's name, home page, background and preview images, language, and other settings. For localization, a skin may also have XML data files containing user-visible strings and names of localized images for the supported languages.

Packaging the Skin Files

After the skin files are created, they must be packaged into a standard Zip format file. Using any tool that can write Zip files, such as PKWare PKZip or WinZip, create a Zip file containing the skin files. A separate Zip file must be produced for each skin under development and these Zip files must be placed in the `C:\Action Engine\Grid\Skins` directory.

Making a Preview Image of the Skin

A preview is similar to a minimal skin with little information. It contains an image representing the appearance of the full skin. It also has the skin identifier that associates the preview with the corresponding skin.

Validating Files and Functionality

As the first level of skin validation, view the skin using the Windows executable `Grid.exe`. This file must be in the `C:\Action Engine\Grid` directory.

Not all of the skin's functionality can be tested in this way; however, the basic navigation using the arrow keys, soft keys, and menus can be tested. To fully validate the skin's display and behavior, it must be installed on the device.

Installing Action Grid on a Device

Action Grid and the skin must be installed and tested on the device to fully validate the correct display and functionality.

Pocket PC or Smartphone Device

To install the Action Grid client on a Pocket PC or Smartphone, copy the corresponding `Grid.cab` to the root `"My Device"` directory, then run `Grid.cab` on the device. This installs the `grid.exe` client into the `\Action Engine\Grid` directory. It will also install a small utility application named `gridwd.exe` whose purpose is the restart `grid.exe` if the user stops it.

Action Grid will automatically start when the device is started. To disable automatic startup, there are potentially two items to modify, depending on the device and its settings.

- Delete the file `\Windows\StartUp\Action Grid`.
- Delete two registry values under `HKEY_LOCAL_MACHINE\init`. The first value is `Launch`*XX*, where *XX* is 99 or the first available number less than 99. The second registry value is `Depend`*XX*.

Symbian Device

To install Action Grid on a Symbian device, copy the corresponding `Grid.sis` to the device using infrared (IR) or Bluetooth. Run the `Grid.sis` file to install Action Grid.

Deploying Action Grid to the Customers' Device

All of Action Engine's products can be installed on customers' devices using a variety of methods, providing a high degree of flexibility and customization. A combination of deployment mechanisms may be used together depending on operator requirements.

An Action Grid deployment must consider both the Action Grid client software and the skins. The client software is larger than one individual skin. New skins can be created and deployed separately after an initial launch.

Installing via ROM

Action Engine can work with device manufacturers and wireless operators to load an Action Grid into the device ROM. This provides an optimal out-of-the-box experience for users because the device is ready to run Action Grid skins at the time of purchase.

There are several other advantages in this approach. Loading Action Grid into ROM means less RAM will be needed to run it, conserving device resources and improving performance. Another benefit to locating Action Engine applications in ROM is that applications remain available after a device hard reset. An over-the-air update may be automatically executed after a hard reset to synchronize the device with the latest product or skin changes.

Installing via Memory Card

One of the fastest methods to deploy Action Grid software is with SD or MMC memory cards. Action Engine can preinstall an operator's selected products on a memory card, adding an installer that automatically initiates when the card is inserted into a device. Operators may choose to have the installer copy the Action Engine software to the

device's RAM or the installer can be configured to run the Action Engine software directly from the memory card, freeing the device RAM for other uses.

Memory card deployments cannot update the device's ROM.

Installing via Computer Connection (Infrared, Cradle, or Bluetooth)

Operators who have already deployed devices without Action Engine software may choose to provide a ROM upgrade to end users via CD-ROM distribution and/or downloadable packages on the Internet. User installations are usually provided as a secondary deployment method because not all users connect their devices to PCs.

Action Engine will work with the device manufacturer and wireless operator to update the device ROM image with the Action Grid software and create a PC-based installation package. When the installation program is executed on a host PC with a connected device, the new ROM will be transferred and installed to that device. If installed by a ROM upgrade, the Action Engine software will remain on the device after a hard reset.

Alternatively, a PC-based installation package can be created without requiring involvement from the device manufacturer. This installer will copy the Action Grid software into the device's RAM, leaving the ROM image untouched. Action Engine can create and deploy RAM installations much faster than ROM installations. RAM installations will not remain on the device after a hard reset.

Installing Over-the-Air

Over-the-air (OTA) updates are an extremely powerful deployment approach providing a high level of flexibility to operators. OTA updates allow operators to update Action Engine software and resources using a device's active GPRS connection, without requiring users to connect devices to host PCs, insert memory cards, or return devices to service centers. OTA updates are only possible if Action Engine's Action Grid or Action Remote client software has been previously installed on a device through another deployment method.

Publishing to the Server

Before a skin can be delivered OTA to a customer device, it must be published to server as described in the *Action Grid Skin Packaging and Installation Guide*.

Pushing the Skin to the Phone

Using OTA updates, carriers may choose to deploy new Action Grid skins automatically to customer devices. OTA updates do not modify the device's ROM and only copy the received files to RAM. If an update is received for an Action Engine product that is already in the device's ROM, a newer version of that product will be copied to RAM. Only the RAM version will launch for all subsequent executions of that product.

4. Creating an Action Grid Skin

This section will focus on the development and implementation of an Action Grid skin, using the sample skins as models. The sample skins are in the directories under `C:\Action Engine\Grid\Skins`.

Designing a Skin

Art Files

The artwork is the most highly visible aspect of a skin, setting the theme for the device's display. The looks that a skin designer can create are limited only by the imagination, usability and platform considerations, screen size, as well as certain business considerations. Action Grid provides a powerful, flexible mechanism for quickly designing and deploying skins with a high "wow" factor. This results in enormous differentiation and personalization potential.

Several formats for art files are supported. One traditional guideline is that continuous tone images, such as photographs, display better using bitmap or JPEG format. The bitmap format usually yields a larger file size than JPEG but it does not lose any image quality. The JPEG format uses a *lossy* compression algorithm that discards information as it compresses files. Noticeable decreases in image quality are seen as *artifacts* when file size is reduced. Graphics editing programs allow the artist to select the amount of compression to control the file size and image quality.

The GIF format is usually best for hard edged and flat color images such as logos, where there is little or no change in tonality or shading. The GIF compression algorithm is lossless but the GIF format supports a maximum of 256 colors. Because of the small screen size of the devices, it is often possible to achieve good visual quality and small file size with continuous tone images using the GIF format. It is worth experimenting with the various formats to achieve the best results.

Although images created for Action Grid must be in rectilinear format, the creative use of transparency will enable designers to create virtually any combination of looks and interactivity. Typically, the artwork will include a background image and button images. The artwork will often include status indicators such as battery level and signal strength indicators, among others.

Phone Platform Considerations

Windows Mobile

- Smartphone
- Pocket PC Phone Edition

The screen size for the Smartphone is 176 pixels wide by 220 pixels tall, while the Pocket PC screen is 240x320 pixels. Normally some portion of the screen is reserved by the operating system for the top and bottom system tray areas, but Action Grid permits use of the entire screen.

Smartphone uses Nina as its default system font. All text is displayed in Nina if no other fonts are installed. To avoid unforeseen results on user devices with no additional fonts installed, it is best to design with this information in mind.

Every Pocket PC comes with two standard fonts installed: Tahoma and Courier. If no other fonts are installed, then all other fonts are converted to the closer of these two options as defined by their font descriptions. Tahoma is a sans serif variable width font and Courier is a fixed-width font with serifs.

Embedded artwork could be used that displays other fonts or other font characteristics, but text would necessarily lose its dynamic properties. Pocket PC Phone Edition combines the standard features and functionality of a Windows Pocket PC with those of a full-featured mobile phone.

Symbian

- Symbian OS v6.1/Series 60 v1.2
- Symbian OS v7.0s/Series 60 v2.0

As noted before, devices running these systems have a screen size of 176x208 pixels, so it has the smallest screen among the supported devices.

Custom font creation with Symbian is limited to variations on its resident proprietary system font. That means it is possible to adjust font size and color, or make it bold or italic. Again, the artwork could contain text in other fonts.

Symbian devices support the following image formats:

- BMP: Bitmap picture
- GIF 87a: Graphics Interchange Format
- GIF 89a: GIF with animation support
- JPEG: Joint Photographic Experts Group
- MBM: EPOC multi-bitmap
- PNG: Portable Network Graphics
- WBMP: Wireless Bitmap Picture
- TIFF-F: Tag Image File Format, F profile for Facsimile.

Image Transparency

In the button and status indicator images, Action Grid can use the color of the top-left pixel as the transparent color. Pixels of that color in the image will not be drawn, allowing the background to show. In the sample images that use transparency, the top-left pixel has the color Magenta, with RGB values of 255, 0, 255 (equivalent to #FF00FF).

Background

The current Action Grid skin determines the default background. This image can be thought of as the "stage" that determines the overall thematic look and feel. It also provides the frame against which all the interactive elements such as icons and *fly-out menus* are designed and used.

The background image size should match the device display. Figure 4-1 shows the screen resolution in pixels of the devices supported by Action Grid.

Device	Screen Resolution (pixels, width x height)
Pocket PC	240 x 320
Smartphone	176 x 220
Symbian Series 60	176 x 208

Figure 4-1 Screen sizes of devices

Efficient workflow can be achieved by using the same naming conventions for similar types of images across all skins. For example, if there is an icon named `phone_btn.gif` common to all skins, then each skin can have a different image, but with the same common name as the others.

Icons

Figure 4-2 shows two sample icon images that contain a large area defining transparency. In this case, it would be appropriate to omit anti-aliasing along the borders of the visible image area.

contacts_btn.gif

contacts_btn_over.gif

Figure 4-2 Example icon images in unselected and selected states

Predefined Status Indicators

Action Grid provides predefined status indicators for

- Battery status
- Signal strength
- New SMS received
- New e-mail received
- Missed phone call
- Appointment due
- Task due.

These indicators are implemented as *image strips*, graphics with multiple images contained in one file. Figure 4-3 illustrates some sample status indicator images. These sample battery and signal strength indicators will always be visible, while the others will only display when active.

Indicator	Sample Image Strip	Indicator State
Battery		Depends on mode: low resolution mode is shown
Signal strength		0 – No signal 1 – Low signal 2 – Medium signal 3 – High signal
SMS		0 – No new SMS 1 – New SMS
E-mail		0 – No new e-mail 1 – New e-mail
Missed call		0 – No missed calls 1 – Missed phone call

Appointment		0 – No appointments due 1 – Appointment due
Task		0 – No tasks due 1 – Task due

Figure 4-3 Example status indicator image strips and indicator states

Layout Options

The possibilities for creating unique skin layouts are nearly limitless. Operators can construct Action Grid skins to leverage their brand messaging using the sample skins supplied by Action Engine, or build their own from scratch. Action Engine has designed and implemented these sample skins based on several common approaches to the UI presentation layer. They should serve only as guides and not as limiting factors.

List

The list layout design uses a vertical or hierarchical stacking display to present interactive icons and elements. Within this format, there are still many variations. This kind of layout is well suited for usability, especially when a large number of items or icons are included in the skin. In these cases, the list layout can make navigation simpler. Unfortunately this may occur at the expense of reducing an icon's image size and affecting the skin's visual appeal.

However, within this kind of format, there are many possibilities for unique and compelling designs. For example, the list may be left, center or right justified; even a combination of all these. Accompanying images can be scaled up or down or placed in unusual positions. Creatively displaying the selected and unselected focus states can be especially effective here.

List sample skin, center aligned Rugby sample skin

Figure 4-4 List view examples

Grid

As the name suggests, the grid layout represents a collection of icon elements arranged in a grid-like pattern both across and down the screen. This layout can lend itself to attractive UI displays featuring combinations of images and text. A common approach with this theme is to arrange button icons three across with a variable number of rows.

Even though the underlying images are rectilinear shaped graphic elements, the look and feel of the presentation layer need not be. For example, the basic button image may be a circular icon on a transparent square. Because the background image shows through outside the circular area, the illusion is given of a series of circular shapes when the full screen is displayed at run time.

Grid layout at run-time

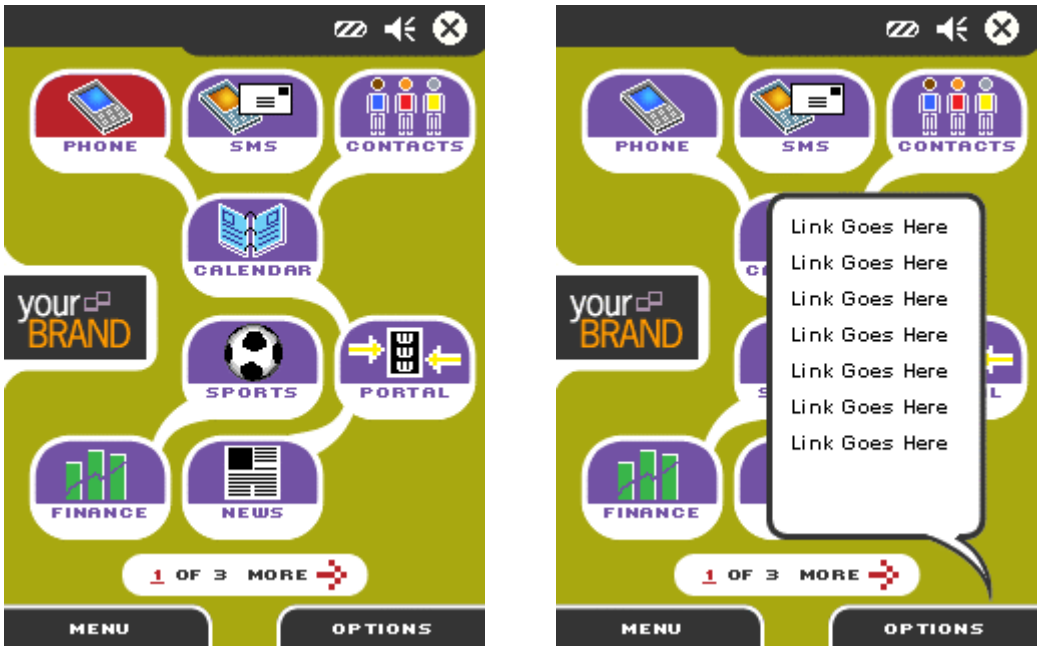
Grid layout showing icon image placement

Figure 4-5 Grid layout example

Freeform

The freeform category of design layouts covers all the other possibilities that do not easily fall into either grid or list. The sample skins shown here were chosen less on the basis of usability than of possibility. This does not imply that usability has been ignored or is unimportant; simply that skin design need not be constrained by conventional approaches or familiar layouts.

The freeform examples are intended as a demonstration of the power and flexibility of the Action Grid solution.



Candy sample: Freeform layout Candy skin showing fly-out menu

Figure 4-6 Freeform layout in the sample Candy skin

Grid.xml

The Action Grid root directory `C:\Action Engine\Grid` contains a file named `grid.xml`. The skin developer can set the `currentSkin` variable to the desired start skin as shown here:

```
<root>
   <var name="defaultSkin">yb</var>
   <var name="background">main_content_bg.gif</var>
   <var name="currentSkin">efc</var>
   <var name="namespace">ae</var>
   <var name="server">http://...</var>
</root>
```

Skin.xml

Each skin contains a file named `skin.xml`. Here is a sample `skin.xml` that shows all the tags in the required order:

```
<skin>
  <title>List</title>
  <author>Action Engine</author>
  <description>List style demonstration skin</description>
  <guid>{E36F9FFF-015D-42a4-BEB4-1D8A2CB2CBDC}</guid>
  <supportedLanguages>en_US en</supportedLanguages>
  <supportedGridVersions>1.0</supportedGridVersions>
  <homepage>home.html</homepage>
  <defaultBackground>main_content_bg.gif</defaultBackground>
  <stringsFile>strings.xml</stringsFile>
```

```
  <imagesFile>images.xml</imagesFile>
  <fallback>en</fallback>
  <language id="de">
    <fallback>es</fallback>
  </language>
  <language id="fr">
    <fallback>de es</fallback>
  </language>
</skin>
```

If a skin is implemented on multiple platforms, the `skin.xml` files for the different platforms will typically be identical.

The `<guid>` element is a *GUID*, a globally unique identifier that is used by Action Grid to identify a skin. This need not vary across implementations of the same skin for different platforms, but it must be distinct for different skins. The GUID of a skin will also be used to create the corresponding skin preview.

The GUID can be created using `guidgen.exe` from Microsoft Visual Studio .NET or any of the GUID creation Web sites. The file `guidgen.exe` is also available from Microsoft at http://www.microsoft.com/downloads/details.aspx?familyid=94551f58-484f-4a8c-bb39-adb270833afc.

The HTML Home Page

The file `home.html` defines the location of each object in the skin. This is where style sheets, status indicators, and icons are specified. Status indicators and icons are positioned using *x,y* pixel coordinates.

Content Considerations

Application Access

Action Grid allows you run any application residing on the device, such as the phone dialer, contacts, calendar, games, and so on. The command line for running an application varies on the different supported devices.

Rollover Animations

There are two methods to show selected and unselected state of the icons. The first method is to use two different images, as shown above. Alternatively, Action Grid can display a focus rectangle around the edge of the selected icon. Finally, both these methods can be used simultaneously.

Using Sounds

The following sample code is from the `home.html` file used in the YourBrand sample skin and demonstrates how to play a sound when an icon is selected.

```
<a href="javascript:playSound('open.wav');shell('tmail.exe');"
    onLeft=  "window.setFocus('sms');"
    onRight= "javascript:playSound('open.wav');shell('tmail.exe');"
    onUp=    "window.setFocus('sports');"
```

```
    onDown=  "window.setFocus('dialer');"
    onFocus=      "sms_btn.src='sms_btn_over.gif';"
    onFocusLeave= "sms_btn.src='sms_btn.gif';"
>
    <img name="sms_btn" src="sms_btn.gif"/>
</a>
```

Sound Formats

The file format used for sounds depends upon the supported formats for each device platform. For example, Windows Smartphone 2002 mobile devices support .Wav-Pcm, 8- or 16-bit mono or stereo and sample rates from 8kHz to 48kHz. In addition, this platform will support MIDI-Only Type 0.

Symbian devices potentially support the following sound/audio formats although .wav files are likely to be of most use here:

- WAV files
- WVE: EPOC native audio file format
- MIDI: Musical Instrument Digital Interface
- AU Audio: usually associated with Sun or Unix
- PCM at 16-bit
- AMR: Adaptive Multi-Rate Codec audio files. Note that AMR is a GSM standard speech coder-decoder that adapts its operation according to channel conditions, operating in both full and half-rate modes. Its primary use is to maximize voice capacity in GSM networks.

Creating HTML Files

Understanding the HTML files

In each sample skin, there are several HTML files. Each skin has a `home.html` that defines the location of each object in the skin. The sample skins also have HTML files that define fly-out menus. A skin may also contain HTML files for other user interface elements. Examples of these additional files include `about.html` in the sample skins, and `__msgbox.html` in the `__grid` skin.

The HTML Home Page

The file `home.html` defines the location of each object in the skin. This is where style sheets, status indicators, and icons are specified. Status indicators and icons are positioned using *x,y* pixel coordinates. The coordinate origin is in the top-left corner. Here is a partial example of a `home.html` page.

The coordinate origin is in the top-left corner. Here is a partial example of a `home.html` page.

```
<html auto-bubble="off">
  <head>
    <link rel="stylesheet" type="text/css" href="home.css" />
```

```
  </head>
  <body>
    <span style="left:200px; top:5px">
      <object name="battery" type="BatteryPlugin" ... />
    </span>
    ...
    <span class="app" name="contacts" style="left:0px; top:80px">
      <a href="javascript:execute('poutlook.exe', 'contacts');"
         onLeft="javascript:setFocus('calendar');"
         onRight="javascript:setFocus('calendar');"
         ...
      >
         <img name="contacts_btn" src="contacts_btn.gif" />
      </a>
    </span>
    ...
  </body>
</html>
```

This example shows several features that will be explained in detail below.

The HTML Element

The `<html>` element is the outermost element of the HTML file. The example shows the custom Action Engine attribute `auto-bubble`, which may be set to `"on"` or `"off"`. Many devices display bubble numbers on links to allow the links to be selected using the keypad. This attribute allows the developer to prevent bubble numbers unless they are specified for a given link.

```
<html auto-bubble="off">
  ...
</html>
```

The Head Element

The `<head>` element specifies the Cascading Style Sheet (CSS) that is used to control the visual formatting of links, menus, and other items in the skin. Supplying additional `<link>` elements will include multiple CSS files. (The CSS file will be covered in a later section.)

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="home.css" />
  </head>
...
```

The Body Element

The `<body>` element declares the body of the HTML document. It can optionally have a `name` element for use in JavaScript code.

```
...
  <body name="body">
    ...
  </body>
```

```
</html>
```

Inside the Body: Div, Span, Object, and Anchor Elements

The and elements are block elements specifying containers for rendering HTML. Using attributes on these elements, styles can be both directly specified and applied from the corresponding CSS file. The primary difference between and elements is that elements by default stretch the entire width of the page, while elements by default are as wide as their contents. In all of these elements, specifying accurate coordinates that match the object size is required to display the skin correctly.

Anchor <a> elements displaying an icon are typically contained within a or . Putting it all together looks like this:

```
...
  <body name="body">
    ...
    <span style="left:201px; top:2px;">
      <object type="BatteryPlugin" width="24" height="17"
              src="battery_anim.gif" params="cell-size:24;transparent:1;"/>
    </span>
    <span style="left:227px; top:3px;">
      <object type="SignalPlugin" width="12" height="17"
              src="signal_anim_full.gif" params="cell-size:12;transparent:1;"/>
    </span>
    <span class="app" name="sms" style="left: 0px; top: 60px; focus: true">
      <a href=    "javascript:playSound('open.wav');
                   shell('windows\\tmail.exe');"
        onLeft=   "window.setFocus('sms');"
        onRight=  "javascript:playSound('open.wav');
                   shell('windows\\tmail.exe');"
        onUp=     "window.setFocus('sports');"
        onDown=   "window.setFocus('dialer');"
        onFocus=       "sms_btn.src='sms_btn_over.gif';"
        onFocusLeave=  "sms_btn.src='sms_btn.gif';"
      >
        <img name="sms_btn" src="sms_btn.gif"/>
      </a>
    </span>
    <span class="app" name="dialer" style="left: 0px; top: 103px;">
      <a href=    "javascript:playSound('open.wav');
                   window.openMenu('menuphone.html', 'menu', 60, 97, 115, 170)"
        onLeft=   "window.setFocus('dialer');"
        onRight=  "javascript:playSound('open.wav');
                   window.openMenu('menuphone.html', 'menu', 60, 97, 115, 170)"
        onUp=     "window.setFocus('sms');"
        onDown=   "window.setFocus('contacts');"
        onFocus=       "dialer_btn.src='phone_btn_over.gif';"
        onFocusLeave=  "dialer_btn.src='phone_btn.gif';"
      >
        <img name="dialer_btn" src="phone_btn.gif"/>
      </a>
    </span>
    ...
  </body>
</html>
```

This portion of the HTML from the YourBrand skin for Pocket PC shows two status indicators, and two anchor elements that display icons and define behavior.

The status indicators are implemented as `<object>` tags with `type` attributes. Other attributes include `src`, `width`, `height`, and `params`. For details on all the tags and attributes, see *Action Grid HTML and JavaScript Reference*

Menu HTML Pages

The sample skins contain HTML pages defining fly-out menus. Like `home.html`, the menu HTML files have an outer `<html>` tag containing `<head>` and `<body>` elements. The `<head>` element can contain a link to a style sheet. Inside the `<body>` element, there are typically anchor `<a>` elements with hyperlinked text within a `<div>` or `<span>`.

Here is the sample phone menu from the YourBrand skin for Pocket PC:

```
<html auto-bubble="off">
  <head>
    <link rel="stylesheet" type="text/css" href="flyoutmenu.css"/>
  </head>
  <body>
    <div class="menu" name="menu">
      <div class="menuitem" name="place_call" style="focus: true">
        <span style="border-left: 2px #FFFFFF;">
          <a href=   "javascript:playSound('tap.wav');
                      shell('Windows\\cprog.exe');window.close();"
            class=   "menuTxt"
            bubble-xoffset="5"
            onLeft=  "window.close();"
            onRight= "window.close();"
            onUp=    "window.setFocus('call_log');"
            onDown=  "window.setFocus('call_log');"
            onFocus=      "this.parent.style.border-left:2px #000000;"
            onFocusLeave= "this.parent.style.border-left:2px #ffffff;"
          >
            <ae:string id="S_PLACE_CALL"/>
          </a>
        </span>
      </div>
      <div class="menuitem" name="call_log">
        <span style="border-left: 2px #FFFFFF;">
          <a href=   "javascript:playSound('tap.wav');
                      launchCallLog();window.close();"
            class=   "menuTxt"
            bubble-xoffset="5"
            onLeft=  "window.close();"
            onRight= "window.close();"
            onUp=    "window.setFocus('place_call');"
            onDown=  "window.setFocus('place_call');"
            onFocus=      "this.parent.style.border-left:2px #000000;"
            onFocusLeave= "this.parent.style.border-left:2px #ffffff;"
          >
            <ae:string id="S_CALL_LOG"/>
          </a>
        </span>
```

```
      </div>
    </div>
  </body>
</html>
```

This fly-out menu uses the style sheet `flyoutmenu.css`. The style sheet specifies the background image and size of the fly-out menu, as well as attributes of the `menu`, `menuitem` and `menuTxt` styles.

This HTML file uses the `ae:string` variable substitution. This code retrieves the string identified by the `id` attribute from the `strings.xml` file, based on the current language and locale settings.

Positioning Elements on the Screen

Elements are positioned on the screen using pixel coordinates starting at the top-left corner of the screen. The position is specified in the HTML files and can also be affected by the settings in the CSS file. Action Grid does minimal verification of the coordinates, so it is necessary to validate that the text and images display as desired.

Creating the CSS Files

Defining Styles for the Skin

Cascading Style Sheets can be used to define additional display styles. The CSS files in Action Grid are very similar to standard CSS files with the addition of an XML `<css>` tag. The file contains named style blocks specifying formatting information such as font name, font size, font and background colors, text positioning, background images, and more. Here is a sample CSS file from the YourBrand skin:

```
<css>
BODY
{
  background-image: "<ae:configvar id="background"/>";
  width: 240px;
  font-size: 8px;
  font-face: tahoma;
  onSoft2: "playSound('open.wav');window.openmenu('menuhome.html',
            'menu', 120, 129, 115, 170);window.setFocus('soft2');";
  focus-rect: false;
}
.app
{
  height: 55px;
  width: 40px;
  transparent: true;
}
.menubutton2
{
  left: 120px;
  top: 300px;
  width: 120px;
  height: 20px;
}
```

```
</css>
```

If the width and height values for a style go beyond the boundary of the device display, images and text will be clipped.

Integrating Content

Icons

To indicate selected and unselected icon states, there are two options. Action Grid can display a highlighted square around the icon, or the icon can be supplied in two forms to represent its selection state.

In icon images, transparency can be defined by the color in the top-left pixel (x,y = 0,0). This allows the background to show through in image areas that contain this color. The color used to denote transparency in the sample skins has RGB values of 255, 0, 255, but any color will work.

Figure 4-7 illustrates two sample icon images in both unselected and selected states. In the two images on the right (Contacts), a large amount of image area used to define transparency. In this case, it would be appropriate to omit anti-aliasing along the borders of the visible image area.

phone_btn.gif phone_btn_over.gif contacts_btn.gif contacts_btn_over.gif

Figure 4-7 Example buttons in unselected and selected states

Phone Services

An Action Grid skin can access the phone functionality of a smart device. For example, here is the sample phone menu from the YourBrand skin for Pocket PC:

```
<div class="menuitem" name="place_call" style="focus: true">
  <span style="border-left: 2px #FFFFFF;">
    <a href="javascript:playSound('tap.wav');
             shell('\\Windows\\cprog.exe');window.close();"
      class=  "menuTxt"
      bubble-xoffset="5"
      onLeft=  "window.close();"
      onRight= "window.close();"
      onUp=    "window.setFocus('call_log');"
      onDown=  "window.setFocus('call_log');"
      onFocus=       "this.parent.style.border-left:2px #000000;"
      onFocusLeave=  "this.parent.style.border-left:2px #ffffff;"
    >
      <ae:string id="S_PLACE_CALL"/>
    </a>
  </span>
</div>
<div class="menuitem" name="call_log">
  <span style="border-left: 2px #FFFFFF;">
```

```
      <a href="javascript:playSound('tap.wav');
               launchCallLog();window.close();"
        class=   "menuTxt"
        bubble-xoffset="5"
        onLeft=  "window.close();"
        onRight= "window.close();"
        onUp=    "window.setFocus('place_call');"
        onDown=  "window.setFocus('place_call');"
        onFocus=       "this.parent.style.border-left:2px #000000;"
        onFocusLeave=  "this.parent.style.border-left:2px #ffffff;"
      >
        <ae:string id="S_CALL_LOG"/>
      </a>
    </span>
  </div>
```

This sample uses the JavaScript `shell` function to run the phone program `\Windows\cprog.exe` on the device. The name of the phone program is device-dependent. To view the call log, the `launchCallLog` function is used. This JavaScript function is supported on Pocket PC devices.

Applications

To run an application on the device, use either the JavaScript `shell` function or the `execute` function. On Symbian devices, the `shell` and `execute` functions are identical. On Pocket PC and Smartphone devices, `shell` internally calls `ShellExecuteEx`, while `execute` internally calls `CreateProcessEx`. Both of these JavaScript functions allow additional command line parameters to be passed. The **Phone Services** section above shows an example using the `shell` function.

Sound Files

To use sounds in an Action Grid skin, the sound file must be included in the .Zip file containing the skin. When an event occurs, the corresponding JavaScript function `playSound` can be called.

Localizing Skins

If a skin going to be deployed in multiple languages, it is mandatory to separate user interface strings and some other UI elements from the core functionality. Action Grid provides methods to localize strings, images, and XML, based on the current device language setting.

Skin.xml

Each skin contains a file named `skin.xml`. The tags used in localizing skins are shown in this `skin.xml` fragment.

```
<skin>
  ...
  <supportedLanguages>en_US en</supportedLanguages>
  ...
```

```
  <stringsFile>strings.xml</stringsFile>
  <imagesFile>images.xml</imagesFile>
  <fallback>en</fallback>
  <language id="de">
    <fallback>es</fallback>
  </language>
  <language id="fr">
    <fallback>de es</fallback>
  </language>
</skin>
```

The `<supportedLanguages>` tag lists the supported languages. There can be multiple language identifiers separated by spaces, inside this tag. The identifiers are ISO 2-letter language codes or language codes plus a regional or sublanguage variation.

The `<fallback>` tag lists the default language that will be used, if there are no other matches. There may be only one `<fallback>` tag.

The data inside the `<stringsFile>` tag is the name of the XML file containing localized text. Similarly, inside the `<imagesFile>` tag is the name of the XML file with names of localized images. In the sample `skin.xml` shown, the strings file is specified as `strings.xml`.

Localized Strings File

The strings file uses `id` attributes to associate a localized string with the corresponding language. The `id` attribute can be the 2-letter language code or the sublanguage variation. A section of a `strings.xml` supporting generic English, US English, English (United Kingdom), Spanish, and French might look like this:

```
<strings>
  <language id="en">
    ...
    <string id="S_EXIT">Exit</string>
    <subLanguage id="US">
      <string id="S_EXIT">Later, dude</string>
    </subLanguage>
    <subLanguage id="GB">
      <string id="S_EXIT">Cheerio!</string>
    </subLanguage>
  </language>
  <language id="es">
    ...
    <string id="S_EXIT">Salir</string>
  </language>
  <langauge id="fr">
    ...
    <string id="S_EXIT">Sortir</string>
  </langauge>
</strings >
```

Using Localized Strings

In the HTML and CSS files that display localized strings, the `<ae:string>` tag is used to extract the correct string based on its `id` attribute and the device language setting. For example:

```
...
  <span style="border-left: 2px #FFFFFF;">
    <a href="javascript:playSound('tap.wav');exit();"
       ...
    >
      <ae:string id="S_EXIT"/>
    </a>
  </span>
...
```

Localized Images File

The localized images file has the same structure as the localized strings file. It uses `id` attributes to associate a localized string with the corresponding language. An `images.xml` supporting English and German might look like this:

```
<images>
  <language id="en">
    <image id="FlyoutPane02">flyout_pane_02.gif</image>
  </language>
  <language id="de">
    <image id="FlyoutPane02">flyout_pane_02_de.gif</image>
  </language>
</images>
```

Using Localized Images

In the HTML and CSS files that display localized images, the `<ae:image>` tag is used to extract the correct image based on its `id` attribute and the device language setting. For example:

```
<css>
BODY
{
  background-image: "<ae:image id="FlyoutPane02"/>";
  ...
```

Other Language-Dependent Settings

The `<ae:localize>`, `<ae:language>`, and `<ae:subLanguage>` tags can be used to localize other content in the HTML and CSS files. For example:

```
<css>
BODY
{
  background-image: "<ae:image id="FlyoutPane02"/>";
  <ae:localize>
    <ae:language id="en">
```

```
      width: 115px;
      <ae:subLanguage id="US">
        width:134px;
      </ae:subLanguage>
    </ae:language>
    <ae:language id="de">
      width: 173px;
    </ae:language>
  </ae:localize>
  transparent: true;
}
...
```

Language and Fallback Settings

The `<fallback>` and `<language>` tags in `skin.xml` affect the method that Action Grid uses to select and display localized content.

The first `<fallback>` tag specifies the language strings that will be used if the current device language setting is not listed in a `<language>` tag. In the example `skin.xml` above, if the device is set to any variant of German, then Action Grid will search first for German content, then Spanish content from the `<fallback>` setting of `es`, and finally English content.

If a `<subLanguage>` tag is present in the localized content, Action Grid will try to provide an exact match with language and sublanguage. However, if there is a match with primary language but the sublanguage is missing or mismatched, then that content will be used.

Capturing Screen Events

An Action Grid responds to various events by providing JavaScript for the event inside the anchor `<a>` tag. In addition to the supported events, a skin developer will also use JavaScript in the `href` attribute to specify an action when the `<a>` link is selected. Events include pressing keys on the keypad, pressing soft keys, rocker or joystick movements, and focus changes.

Supported Events

onBack

Specifies the action to take when the user presses the “Back” key. This is only supported on Smartphone devices.

onDown

Specifies the action to take when the down arrow key, rocker, or joystick is pressed.

onFocus

Specifies the action to take when the focus enters.

onFocusLeave

Specifies the action to take when the focus leaves.

onLeft

Specifies the action to take when the left arrow key, rocker, or joystick is pressed.

onRight

Specifies the action to take when the right arrow key, rocker, or joystick is pressed.

onSoft1

Specifies the action to take when the first soft key is pressed.

onSoft2

Specifies the action to take when the second soft key is pressed.

onUp

Specifies the action to take when the up arrow key, rocker, or joystick is pressed.

Creating Skin Previews

A skin preview contains an image representing the appearance of the full skin and the skin identifier that associates the preview with the corresponding skin.

Skin.xml

A skin preview will contain a `skin.xml` file with a different set of tags than the `skin.xml` for the full skin. The `<guid>` tag in the preview must match the `<guid>` tag in the full skin.

```
<skin-preview>
  <title>Your Brand</title>
  <author>Action Engine Corporation</author>
  <description>A preview for Your Brand</description>
  <guid>{E36F9FFF-015D-42a4-BEB4-1D8A2CB2CBDC}</guid>
  <preview>yb.gif</preview>
</skin-preview>
```

Skin Preview Image

The preview image will show the full skin's appearance as shown in Figure 4-8.

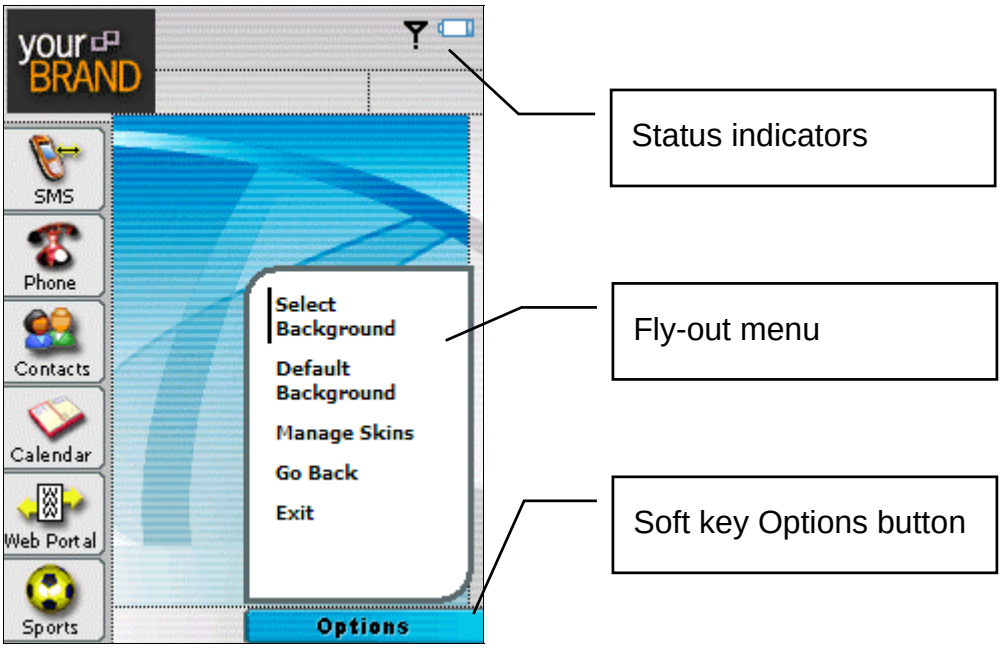


Figure 4-8 The preview image

Packaging the Skin Preview

Use your Zip utility to package the two files into a compressed .Zip file.

5. Testing Skin Form and Functionality

Editing the Grid.xml File

During development and validation of Action Grid skins, it may be useful to edit `grid.xml` to specify a different server name or default skin.

Using the Grid.exe Desktop Client

After an Action Grid is packaged in a Zip file, and the file is placed in `C:\Action Engine\Grid\Skins`, then testing and validation may begin. As a first step in validating the skin's display and functionality, view the skin in the Windows desktop client. The executable file `grid.exe` is located in `C:\Action Engine\Grid`.

To display a Pocket PC emulator with 240 x 320 pixels in screen size, run `grid.exe` with no command line parameters. To display a Smartphone emulator with 176 x 220

pixels, use the command line `grid.exe 176 220`. To display a Symbian device emulator with 176 x 208 pixels, use the command line `grid.exe 176 208`.

Soft Keys

Use the `F1` and `F2` keys to simulate the soft keys.

Validation

Using the `grid.exe` emulator, it is possible to test the visual display and some of the behavior of a skin. Items that can be tested include

- Background image display
- Button and icon images
- Display of fly-out menus
- Navigation using arrow keys
- Navigation using soft keys
- Navigation using a pointer, for Pocket PC.

Limitations

Using the `grid.exe` emulator, some functionality cannot be tested, including

- Invoking most applications, such as the Phone, Contacts, Inbox, or Calendar
- Some status indicators, such as signal strength and SMS.

Installing a Skin on a Phone

Because of the significant functional differences between the Windows desktop client emulator and an actual device, it is imperative to install and validate an Action Grid skin on the device.

Bluetooth/Cradle/IR

To install the Action Grid client and sample skins on a Pocket PC or Smartphone, copy the corresponding `grid.cab` to the device and then run the file. To copy the file, use any available connection mechanism: Bluetooth, a cradle, or infrared. After this step is completed, a new custom skin may be copied to the `\Action Grid\Skins` directory on the device.

To install the Action Grid client and sample skins on a Symbian device, copy `grid.app` and `grid.sis` to the device and then run the file. After this step is completed, a new custom skin may be copied to the `\Action Grid\Skins` directory on the device.

6. Action Grid HTML and JavaScript Reference

HTML Tag Information

Tags Supported in Action Grid

Action Grid supports most of the standard HTML tags. Here is the list of supported tags:

A	DIV	IMG	OL	U
AREA	FONT	LI	P	UL
B	HR	LINK	SPAN	
BODY	HTML	MAP	STYLE	
BR	I	OBJECT	TITLE	

Object Tag

The `<object>` tag is used for status indicators. The status indicator will automatically display the corresponding section of the image strip based on the indicator state. Attributes for the `<object>` tag, including the `type` attribute value shown in Figure 6-1, are case-sensitive.

Indicator	type Attribute Value	Indicator State
Appointment	`AppointmentDuePlugin`	0 – No appointments due 1 – Appointment due
Battery	`BatteryPlugin`	Depends on mode: low resolution has 3 states and high resolution has 10 states
E-mail	`NewEmailPlugin`	0 – No new e-mail 1 – New e-mail
Missed call	`MissedCallPlugin`	0 – No missed calls 1 – New missed phone call
Signal strength	`SignalPlugin`	0 – No signal 1 – Low signal 2 – Medium signal 3 – High signal
SMS	`NewSmsPlugin`	0 – No new SMS 1 – New SMS
Task	`TaskDuePlugin`	0 – No tasks due 1 – Task due

Figure 6-1 Status indicator types and indicator states

AppointmentDuePlugin

The `AppointmentDuePlugin` object displays the status of Calendar appointments, showing whether or not an appointment is due. An appointment will be marked as due from the time of its reminder or its start time, until the end of the appointment.

Example

```
<span style="left:163px; top:2px;">
  <object type="AppointmentDuePlugin" width="17" height="17"
```

```
            src="appt_anim.gif" params="cell-size:17;transparent:1;"/>
  </span>
```

AdvBatteryPlugin

The `AdvBatteryPlugin` object displays the battery status with a detailed level of granularity.

0 – Undetermined
1 – On charger
2 – Empty battery
3 – Very low battery
4 – Low battery
5 – Medium-low battery
6 – Medium battery
7 – Medium-high battery
8 – High battery
9 – Full battery

Figure 6-2 Battery status indicator states and sample image

Example

```
  < span style="left:201px; top:2px;">
    <object type="BatteryPlugin" width="24" height="17"
       src="battery_anim_ex.gif" params="cell-size:24;transparent:1;"/>
  </span>
```

BatteryPlugin

The `BatteryPlugin` object displays the battery status with a basic level of granularity.

0 – Low battery
1 – Medium battery
2 – Full battery

Figure 6-3 Battery status indicator states and sample image

Example

```
  < span style="left:201px; top:2px;">
    <object type="BatteryPlugin" width="24" height="17"
       src="battery_anim.gif" params="cell-size:24;transparent:1;"/>
  </span>
```

NewEmailPlugin

The `NewEmailPlugin` object shows whether or not there is a new unread e-mail message.

Example

```
<span style="left:182px; top:0px;">
  <object type="NewEmailPlugin" width="17" height="17"
          src="new_email_status_anim.gif"
          params="cell-size:17; transparent:1;"/>
</span>
```

MissedCallPlugin

The `MissedCallPlugin` object shows whether or not there is a missed phone call.

Example

```
<span style="left:182px; top:0px;">
  <object type="MissedCallPlugin" width="17" height="17"
          src="p_missedCall.gif"
          params="cell-size:17; transparent:1;"/>
</span>
```

SignalPlugin

The `SignalPlugin` object shows the current signal strength.

Example

```
<span style="left:227px; top:3px;">
  <object type="SignalPlugin" width="12" height="17"
          src="signal_anim_full.gif"
          params="cell-size:12; transparent:1;"/>
</span>
```

NewSmsPlugin

The `NewSmsPlugin` object shows whether or not there is a new unread SMS message.

Example

```
<span style="left:125px; top:2px;">
  <object type="NewSmsPlugin" width="17" height="17"
          src="sms_anim.gif" params="cell-size:17;transparent:1;"/>
</span>
```

TaskDuePlugin

The `TaskDuePlugin` object shows whether or not there is a task due.

Example

```
<span style="left:144px; top:2px;">
  <object type="TaskDuePlugin" width="17" height="17"
          src="tasks_anim.gif" params="cell-size:17;transparent:1;"/>
</span>
```

Other Objects

In addition to the status indicator, Action Grid provides several other convenient user interface elements.

Object	type Attribute Value
Animation	AnimationPlugin
Progress bar	ProgressBar
Scroll bar	ScrollBar
Scroll indicator	ScrollIndicator
Time	TimePlugin

Figure 6-4 Additional provided object types

AnimationPlugin

The AnimationPlugin object displays an animation by repeatedly cycling through cells in an image strip. The required params attribute must include a bitmap image strip file name and a cell-size value to specify the width in pixels of a cell or frame in the image strip. Optional values are the spin-rate in milliseconds (default is 100) and transparent (0 or 1, default is 0).

The type="AnimationPlugin", width, and height attributes are required. If the required attributes are not supplied, the animation will not be displayed.

Example

```
<span style="left:124px; top:6px;">
  <object type="AnimationPlugin" width="11px" height="18px"
          params="cell-size:11; bitmap:scroll-indicator.gif;
                  spin-rate:200; transparent:1"/>
</span>
```

ProgressBar

The ProgressBar object displays a progress bar. This is used in the file __progress.html in the standard __grid skin.

Example

```
<div class="progressBar">
  <object name="progress" type="ProgressBar"
          width="134px" height="12px"
          params="bitmap:__progress.gif; max-value:100;"/>
</div>
```

ScrollBar

The ScrollBar object displays a customized scroll bar. Bitmaps and colors are specified using the required params attribute.

The type="ScrollBar", width, and height attributes are required. If the required attributes are not supplied, the animation will not be displayed.

Examples

```
<!--  An example scroll bar using custom bitmaps  -->
```

```
<div style="left:220px; top:50px; position: float;">
  <object type="ScrollBar" width="10" height="200"
          params="up-bitmap: up.gif; down-bitmap: down.gif;
                  indicator-bitmap: ind.gif;
                  background-color: #A0A0A0; border-color: #000000;
                  auto-hide: 1;"/>
</div>

<!--  An example scroll bar using custom colors  -->
<div style="left:220px; top:50px; position: float;">
  <object type="ScrollBar" width="10" height="200"
          params="indicator-color: #FF0000;
                  background-color: #000000;
                  border-color: #00FFFF;"/>
</div >
```

ScrollIndicator

The `ScrollIndicator` object displays a scrolling indicator. Like the status indicators, the image is implemented as an image strip. Action Grid displays a section of the image strip corresponding to the scrolling position of the page. Figure 6-4 shows a sample scrolling indicator image strip with each cell 11 pixels wide.

Figure 6-5 Sample scrolling indicator image strip

Example

```
<div style="left:115px; top:290px; position: float;">
  <object type="ScrollIndicator" width="11" height="15"
          params="cell-width: 11; bitmap: scroll-indicator.gif;"/>
</div>
```

TimePlugin

The `TimePlugin` object displays the current time and date using the formats specified by device settings. For example, the time and date might show as “11:27 AM 6/17/2004”. Because it displays text, the `object` tag will often be contained within a `div` or `span` that specifies text properties such as font name, font size, and color.

The `name` attribute is optional. If it is provided, then the specified name can be used to manipulate the object with JavaScript.

The `type="TimePlugin"`, `width`, and `height` attributes are required. If the required attributes are not supplied, the time will not be displayed. To prevent the date and time from being clipped, the object must be large enough to display the text in the font specified by the `div` or `span` containing the object.

Example

```
<span class="timeDisplay" style="left:144px; top:6px;" >
  <object name="time" type="TimePlugin" width="100" height="15"/>
</span>
```

Variables in HTML

The HTML supported in Action Grid has a set of variables evaluated at run-time to aid the skin developer with localization and other tasks. The variable names are HTML tags prefixed with `"ae:"`. The tags are block tags, and many of them require an `id` attribute.

ae:buildNumber

The `ae:buildNumber` variable retrieves a build number from Action Grid. In the sample skins, this variable is used in `about.html` to display version and build information.

Example

```
  <div>
    <ae:string id="S_VERSION"/>:
    <ae:majorVersion/>.<ae:minorVersion/>.<ae:buildNumber/>
  </div>
  <div>
    <ae:string id="S_SERVICE_PACK"/>: <ae:servicePack/>
  </div>
```

ae:configvar

The `ae:configvar` variable retrieves a value from `skin.xml`. In the sample skins, this variable is used in `home.css` to retrieve the default background image.

Attributes

`id`

The identifier of the value to retrieve.

Example

```
<css>
BODY
{
  background-image: "<ae:configvar id="background"/>";
  width: 240px;
  font-size: 8px;
  font-face: tahoma;
  onSoft2: "playSound('open.wav');
           window.openmenu('menuhome.html','menu', 120, 129, 115, 170);
           window.setFocus('soft2');";
  focus-rect: false;
}
...
```

ae:image

The `ae:string` variable retrieves a string from the XML images list file specified in `skin.xml`. In the sample skins, this XML strings file is named `images.xml`. The string will be based on the device language settings and the settings in the `skin.xml` file.

Attributes

`id`

The identifier of the image name to retrieve.

Example

```
<css>
BODY
{
  background-image: "<ae:image id="FlyoutPane02"/>";
...
```

ae:language

The `ae:language` variable specifies the language setting of a block of localized content in an HTML or CSS file.

Attributes

`id`

The 2-letter language identifier.

Example

```
<css>
BODY
{
  background-image: "<ae:image id="FlyoutPane02"/>";
  <ae:localize>
    <ae:language id="en">
      width: 115px;
      <ae:subLanguage id="US">
        width:134px;
      </ae:subLanguage>
    </ae:language>
    <ae:language id="de">
      width: 173px;
    </ae:language>
  </ae:localize>
  transparent: true;
}
...
```

ae:localize

The `ae:localize` block specifies a section of HTML or CSS that will be localized based on the device language settings and the `<language>` and `<fallback>` tags in `skin.xml`. Inside the `ae:localize` block, there will be `ae:language` blocks that contain the localized content.

Attributes

`id`

The 2-letter language identifier.

Example

See the example for `ae:language`.

ae:majorVersion

The `ae:majorVersion` variable retrieves a major version number from Action Grid. In the current version of Action Grid, this returns **`1`**.

Example

See the example for `ae:buildNumber`.

ae:minorVersion

The `ae:minorVersion` variable retrieves a minor version number from Action Grid.

Example

See the example for `ae:buildNumber`.

ae:servicePack

The `ae:servicePack` variable retrieves a service pack number from Action Grid.

Example

See the example for `ae:buildNumber`.

ae:skinlist

The `ae:skinlist` variable retrieves a list of skins or previews that are present on the device. In the sample skins, this variable is used in the `__grid` skin in the file `__skinlist.html`.

Attributes

`id`

This attribute must be set to `"skins"` or `"skin-previews"`.

Example

```
<div class="header">
  <ae:string id="S_INSTALLED_SKINS"/>
</div>
<div class="spacer" />
<ul class="skinlist">
  <ae:skinlist id="skins"/>
</ul>
<div class="header">
  <ae:string id="S_PREVIEW_SKINS"/>
</div>
<div class="spacer" />
<ul class="skinlist">
  <ae:skinlist id="skin-previews"/>
</ul>
```

ae:string

The `ae:string` variable retrieves a string from the XML strings file specified in `skin.xml`. In the sample skins, this XML strings file is named `strings.xml`.

Attributes

`id`

The identifier of the string to retrieve.

Example

```
  <a href=   "javascript:playSound('tap.wav');
              shell('Windows\\cprog.exe');window.close();"
    class=   "menuTxt"
    bubble-xoffset="5"
    onLeft=  "window.close();"
    onRight= "window.close();"
    onUp=    "window.setFocus('call_log');"
    onDown=  "window.setFocus('call_log');"
    onFocus=      "this.parent.style.border-left:2px #000000;"
    onFocusLeave= "this.parent.style.border-left:2px #ffffff;"
  >
    <ae:string id="S_PLACE_CALL"/>
  </a>
```

ae:subLanguage

The `ae:subLanguage` variable specifies the language setting when retrieving content using `ae:string` or `ae:image`.

Attributes

`id`

The 2-letter sublanguage identifier.

Example

```
<css>
BODY
{
  background-image: "<ae:image id="FlyoutPane02"/>";
  <ae:localize>
    <ae:language id="en">
      width: 115px;
      <ae:subLanguage id="US">
        width:134px;
      </ae:subLanguage>
    </ae:language>
    <ae:language id="de">
      width: 173px;
    </ae:language>
  </ae:localize>
  transparent: true;
}
...
```

JavaScript Reference

Windows, styles, sounds, and other elements of an Action Grid skin can be manipulated using JavaScript in HTML and style sheets.

Scripting Objects

Nodes and Styles

Some scripting functionality modifies an object style. Each item in an HTML file is a *node*. Nodes include anchor `<a>` elements, and `<div>` and `<span>` elements. A node can be either named using the `id` or `name` attribute, or unnamed. To reference a node, use the name or a relative reference. Relative references can be the current node or a parent node.

this

The word `this` returns the current node.

parent

The word `parent` returns the parent, or container, of the current node.

Example

To modify the background color style of the parent node:

```
this.parent.style.background-color = #FFFFFF;
```

Windows

Some scripting functionality, such as the `setFocus` function, requires a reference to a window. A window can be either named by the code that opens the window, or unnamed. To reference a window, use the name or a relative reference. Relative references can be the current window or the next window on the z-order stack.

home

The word `home` returns the skin's main, or top, window.

window

The word `window` returns the current window.

parent

The word `window` returns the next window on the z-order stack.

Example

Here is an example that uses the `setFocus` and `openMenu` functions:

```
<a href=    "javascript:playSound('open.wav');
             window.openMenu('menuphone.html', 'menu', 60, 97, 115, 170)"
 onLeft=    "window.setFocus('dialer');"
 onRight=   "javascript:playSound('open.wav');
             window.openMenu('menuphone.html', 'menu', 60, 97, 115, 170)"
 onUp=      "window.setFocus('sms');"
 onDown=    "window.setFocus('contacts');"
 onFocus=       "dialer_btn.src='phone_btn_over.gif';"
 onFocusLeave=  "dialer_btn.src='phone_btn.gif';"
>
  <img name="dialer_btn" src="phone_btn.gif"/>
</a>
```

Script Parameters

Many of the scripting functions require one or more parameters. The parameters are integers for coordinate values (`x`, `y`, `width`, `height`) and strings for almost all other values. The strings must be enclosed in single quotes (`'`).

Scripting Functions

close

The `close` function closes a window. It requires a reference to a window.

Parameters

This function has no parameters.

Example

```
<a href=   "javascript:playSound('tap.wav');
            shell('Windows\\cprog.exe');window.close();"
  class=   "menuTxt"
  bubble-xoffset="5"
  onLeft=  "window.close();"
  onRight= "window.close();"
  onUp=    "window.setFocus('call_log');"
  onDown=  "window.setFocus('call_log');"
  onFocus=      "this.parent.style.border-left:2px #000000;"
  onFocusLeave= "this.parent.style.border-left:2px #ffffff;"
>
  <ae:string id="S_PLACE_CALL"/>
</a>
```

execute

The `execute` function sends a command with an optional parameter to the device operating system. This function is primarily intended for use on Symbian devices. The `shell` function is recommended on Smartphone and Pocket PC devices.

Parameters

`app-document`

String: The application or document to run.

`command-line`

String: Optional. Additional command line parameter.

Example

```
<a href=    "javascript:playSound('open.wav');execute('mce.app');"
  onLeft=   "window.setFocus('sms');"
  onRight=  "javascript:playSound('open.wav');execute('mce.app');"
  onUp=     "window.setFocus('sports');"
  onDown=   "window.setFocus('dialer');"
  onFocus=       "sms_btn.src='sms_btn_over.gif';"
  onFocusLeave=  "sms_btn.src='sms_btn.gif';"
>
  <img name="sms_btn" src="sms_btn.gif"/>
</a>
```

exit

The `exit` function quits Action Grid.

Parameters

This function has no parameters.

Example

```
<a ref="javascript:playSound('tap.wav');exit();"
  class=    "menuTxt"
  bubble-xoffset= "5"
  onLeft=   "window.close();"
  onRight=  "window.close();"
  onUp=     "window.setFocus('about');"
  onDown=   "window.setFocus('set_bg');"
  onFocus=       "this.parent.style.border-left:2px #000000;"
  onFocusLeave=  "this.parent.style.border-left:2px #ffffff;"
>
  <ae:string id="S_EXIT"/>
</a>
```

hide

The `hide` function hides the Action Grid window.

Parameters

This function has no parameters.

Example

```
<a ref="javascript:playSound('tap.wav');hide();"
  class=    "menuTxt"
  bubble-xoffset= "5"
  onLeft=   "window.close();"
  onRight=  "window.close();"
  onUp=     "window.setFocus('about');"
  onDown=   "window.setFocus('set_bg');"
  onFocus=       "this.parent.style.border-left:2px #000000;"
  onFocusLeave=  "this.parent.style.border-left:2px #ffffff;"
>
  <ae:string id="S_EXIT"/>
</a>
```

launchCallLog

The `launchCallLog` function runs the call log program on a Pocket PC device. Other devices have a separate executable that can be run using the `shell` or `execute` function.

Parameters

This function has no parameters.

Example

```
<a href="javascript:playSound('tap.wav');launchCallLog();
        window.close();"
  class=    "menuTxt"
  bubble-xoffset="5"
  onLeft=   "window.close();"
```

```
    onRight=  "window.close();"
    onUp=     "window.setFocus('place_call');"
    onDown=   "window.setFocus('place_call');"
    onFocus=       "this.parent.style.border-left:2px #000000;"
    onFocusLeave=  "this.parent.style.border-left:2px #ffffff;"
  >
    <ae:string id="S_CALL_LOG"/>
  </a>
```

loadSkin

The `loadSkin` function closes all windows in the current Action Grid skin and loads a specified new skin. To allow the user to select from all the available skins installed on the device, use the `manageSkins` function instead.

Parameters

`skin-guid`

String: The GUID of the skin, as specified in the skin's `skin.xml` file. The skin's `.Zip` file must be in the device's `skins` directory.

Calling the `loadSkin` function with a zero-length string reloads the current skin.

`download-skin`

Integer: Optional; defaults to 0 if not provided. If this parameter is non-zero, then Action Grid will attempt to download the skin from the server if it is not present on the device.

Example

```
  <a href="javascript:loadSkin('{E36F9FFF-015D-42a4-BEB4-1D8A2CB2CBDC}')">
    Load the YourBrand skin
  </a>
```

manageSkins

The `manageSkins` function displays a skin chooser screen allowing the user to select from the currently installed skin and skin previews on the device.

Parameters

This function has no parameters.

Example

```
  <a  href="javascript:playSound('open.wav');
             manageSkins();window.close();"
    class=    "menuTxt"
    bubble-xoffset="5"
    onLeft=   "window.close();"
    onRight=  "window.close();"
    onUp=     "window.setFocus('def_bg');"
    onDown=   "window.setFocus('go_back');"
    onFocus=       "this.parent.style.border-left:2px #000000;"
    onFocusLeave=  "this.parent.style.border-left:2px #ffffff;"
  >
    <ae:string id="S_MANAGESKINS"/>
  </a>
```

messageBox

The `messageBox` function displays a message box. The visual display of the message box is device-specific.

Parameters

`title`

String: The message box title.

`text`

String: The message box text.

`button1`

String: Optional. Identifies the text in the first button. If it is not provided, `'AEMB_OK'` is the default.

The value of this string will cause Action Grid to look up and display a string from the skin's `strings.xml` file. The `button1` parameter must be one of these values:

```
'AEMB_OK'        'AEMB_YES'     'AEMB_NO'
'AEMB_CANCEL'    'AEMB_RETRY'
```

`button2`

String: Optional. Identifies the text in the second button. If it is not provided, a second button will not be displayed. If this parameter is provided, it must be one of the values listed in `button1`.

Example

```
<a href="javascript:messageBox('Title', 'Message text', 'AEMB_OK');">
  Show message box
</a>
```

open

The `open` function opens a new window based on an HTML file. The `open` function is typically used for a full screen HTML file. It requires a reference to a window.

The `open` function always creates a new window. To load a different HTML file into an existing window, use `setHRef`.

Parameters

`html-file`

String: An HTML file that defines the new window.

`name`

String: Optional. A window name for use in scripting. If a name is provided, it can be used by subsequent JavaScript functions to refer to this window.

`x`

Integer: Optional. The x coordinate in pixels of the top-left corner of the window. If none of the coordinates are provided, the window is assumed to be full screen. If the window is not full screen, all four coordinates–x, y, width, and height– should be provided.

y

Integer: Optional. The y coordinate in pixels of the top-left corner of the window.

width

Integer: Optional. The width of the window in pixels.

height

Integer: Optional. The height of the window in pixels.

Example

```
<a class="menuTxt"
  href=  "javascript:window.open('test.html');window.close();"
  bubble-xoffset="5"
  onLeft=   "window.close();"
  onRight=  "window.close();"
  onUp=     "window.setFocus('exit');"
  onDown=   "window.setFocus('set_bg');"
  onFocus=       "this.parent.style.border-left:2px #000000;"
  onFocusLeave=  "this.parent.style.border-left:2px #ffffff;"
>
  <ae:string id="S_TEST"/>
</a>
```

openMenu

The openMenu function opens a new window based on an HTML file. The openMenu function is typically used for a fly-out menu that is not full screen. A window created with the openMenu function will close on a Pocket PC when the user taps outside of it. This function requires a reference to a window.

The openMenu function always creates a new window. To load a different HTML file into an existing window, use setHRef.

Parameters

html-file

String: An HTML file that defines the new window.

name

String: Optional. A window name for use in scripting. If a name is provided, it can be used by subsequent JavaScript functions to refer to this window.

x

Integer: Optional. The x coordinate in pixels of the top-left corner of the window. If none of the coordinates are provided, the window is assumed to be full screen. If the window is not full screen, all four coordinates–x, y, width, and height– should be provided.

`y`

Integer: Optional. The `y` coordinate in pixels of the top-left corner of the window.

`width`

Integer: Optional. The width of the window in pixels. If an image is used for the fly-out menu background, this `width` value should match the image width.

`height`

Integer: Optional. The height of the window in pixels. If an image is used for the fly-out menu background, this `height` value should match the image height.

Example

```
<a href=   "javascript:playSound('open.wav');
          window.openmenu('menuphone.html', 'menu', 60, 97, 140, 92);"
  onLeft=  "window.setFocus('dialer');"
  onRight= "javascript:playSound('open.wav');
          window.openmenu('menuphone.html', 'menu', 60, 97, 140, 92);"
  onUp=    "window.setFocus('sms');"
  onDown=  "window.setFocus('contacts');"
  onFocus= "dialer_btn.src='phone_btn_over.gif';"
  onFocusLeave=  "dialer_btn.src='phone_btn.gif';"
>
  <img name="dialer_btn" src="phone_btn.gif"/>
</a>
```

playSound

The `playSound` function plays a sound file.

Parameters

`sound-file`

String: A sound file.

Example

```
<a href="javascript:playSound('tap.wav');launchCallLog();
          window.close();"
  class=    "menuTxt"
  bubble-xoffset="5"
  onLeft=   "window.close();"
  onRight=  "window.close();"
  onUp=     "window.setFocus('place_call');"
  onDown=   "window.setFocus('place_call');"
  onFocus=       "this.parent.style.border-left:2px #000000;"
  onFocusLeave=  "this.parent.style.border-left:2px #ffffff;"
>
  <ae:string id="S_CALL_LOG"/>
</a>
```

previewSkin

The `previewSkin` function displays a preview of an Action Grid skin.

This function is rarely called directly by the skin developer. To view the available skins and previews, use the `manageSkins` function.

Parameters

`skinpreview-guid`

String: The GUID of the preview skin, as specified in the skin's `skin.xml` file. The skin's `.Zip` file must be in the device's `\Action Engine\Grid\previews` directory.

Example

```
<a href="javascript:previewSkin('{E36F9FFF-015D-42a4-BEB4-1D8A2CB2CBDC}');>
  Preview the YourBrand skin
</a>
```

selectBackground

The `selectBackground` function displays a screen to select a custom background image. This screen is defined in the file `__filechooser.html` in the `__grid` skin.

Parameters

This function has no parameters.

Example

```
<a  href=  "javascript:playSound('open.wav');selectBackground();
           window.close();"
  class=   "menuTxt"
  bubble-xoffset="5"
  onLeft=  "window.close();"
  onRight= "window.close();"
  onUp=    "window.setFocus('exit');"
  onDown=  "window.setFocus('def_bg');"
  onFocus=       "this.parent.style.border-left:2px #000000;"
  onFocusLeave=  "this.parent.style.border-left:2px #ffffff;"
>
  <ae:string id="S_SELECT_BACKGROUND"/>
</a>
```

setFocus

The `setFocus` function displays a screen to sets the focus to the named target node. This function requires a reference to a window.

Parameters

`node`

String: The name of the node to set the focus to. This can be an anchor `<a>` element, a `<div>` element, or a `<span>` element.

Example

```
<a  href=  "javascript:playSound('open.wav');selectBackground();
           window.close();"
  class=   "menuTxt"
  bubble-xoffset="5"
  onLeft=  "window.close();"
  onRight= "window.close();"
  onUp=    "window.setFocus('exit');"
  onDown=  "window.setFocus('def_bg');"
```

```
    onFocus=       "this.parent.style.border-left:2px #000000;"
    onFocusLeave=  "this.parent.style.border-left:2px #ffffff;"
  >
    <ae:string id="S_SELECT_BACKGROUND"/>
  </a>
```

setHRef

The `setHRef` function replaces the contents of a window with the specified HTML file. The `setHRef` function can only be called from an existing window reference.

To create a new window, use either `open` or `openMenu`.

Parameters

`html-file`

String: An HTML file that defines the new contents of a window.

Example

```
  <a href="javascript:window.setHRef('newPage.html');>
    Load a different page in this window.
  </a>
```

shell

The `shell` function sends a command with an optional parameter to the device operating system. On Symbian devices, the `shell` and `execute` functions are identical. On Pocket PC and Smartphone devices, `shell` internally calls `ShellExecuteEx`, while `execute` internally calls `CreateProcessEx`. The `shell` function is recommended on Smartphone and Pocket PC devices.

Parameters

`file`

String: The name of the file to open. This can be a document or an executable file.

`command-line`

String: Optional. Additional command line parameters.

Example

```
  <a href="javascript:window.close();shell('http://www.espn.com');"
     class=    "menuTxt"
     bubble-xoffset="7"
     onLeft=   "window.close();"
     onRight=  "window.close();"
     onUp=     "window.setFocus('bbcsports');"
     onDown=   "window.close();"
     onFocus=       "this.style.color=#FF0000;"
     onFocusLeave=  "this.style.color=#000000;"
  >
    ESPN Sports
  </a>
```

syncSkins

The `syncSkins` function synchronizes the skins and previews present on the device with the skins and previews in the catalog on the server. If a skin or preview is present in the server's catalog but not on the device, then the skin or preview will be downloaded to the device over-the-air.

Parameters

`load-manageskins`

Integer: Optional; defaults to 0 if not provided. If this parameter is 0, then Action Grid will reload and display the current skin after synchronization with the server is complete. If it is non-zero, then the skin chooser screen invoked by the `manageSkins` function will be displayed after synchronization.

Example

```
<a href="javascript:window.close();syncSkins(1);">
  Synchronize skins
</a>
```

useDefaultBackground

The `useDefaultBackground` function loads the default background image for the current skin, as defined in `skin.xml`. Any custom background image setting is discarded.

Parameters

This function has no parameters.

Example

```
<a href="javascript:playSound('open.wav');
        useDefaultBackground();window.close();"
   class=   "menuTxt"
   bubble-xoffset="5"
   onLeft=  "window.close();"
   onRight= "window.close();"
   onUp=    "window.setFocus('set_bg');"
   onDown=  "window.setFocus('choose');"
   onFocus=       "this.parent.style.border-left:2px #000000;"
   onFocusLeave=  "this.parent.style.border-left:2px #ffffff;"
>
  <ae:string id="S_DEFAULT_BACKGROUND"/>
</a>
```

Appendix A: Glossary

The following terms are used to describe Action Grid. Familiarity with this terminology will be helpful in understanding the how the Action Grid works.

- **Action Grid** – A software product that provides mobile operators with the ability to deliver a branded User Experience Layer on smart phones. It consists of a client application installed on the device and a server component for deployment and administration.
- **Action Grid skin** – A skin is highly customizable user interface layer for launching applications, displaying status indicators, and providing optimized navigation. A skin defines numerous elements of the user interface, including screen background, layout, icons, labels, menu hierarchies, selected state behaviors, view styles, and colors. Optionally, it can also system sounds.
- **AEHTML** – Action Engine HTML. This is a variant of HTML that supports additional tags for Action Grid.
- **Artifact** – Visible image quality degradation due to excessive JPEG compression. The effects include areas of similar colors being made the same color, sharp edges being blurred, and incorrect colors.

- **Fly-out menu** – A window that can be displayed when the user click an icon in a skin. It is populated with hyperlinked text in list format.
- **Grid.xml** – A file contains the current skin name, background, and other settings. It is located at the `C:\Action Engine\Grid` level in the directory structure.
- **GUID** – Globally Unique Identifier, used by Action Grid to identify skins and skin previews.
- **Icon** – An image in an Action Grid skin that is selected by the user to display a menu, launch an application, or open a file.
- **Image strip** – An image file containing multiple frames or cells, used by a status indicator. Action Grid will automatically display the correct frame based on the indicator's current state.
- **Lossy compression** – File compression that discards information to reduce file size. An example is the JPEG image algorithm.
- **Node** – A scriptable item in an HTML file in a skin. Nodes can be named or unnamed.
- **SD card** – A small removable general-purpose read-write memory card.
- **Skin Preview** – A minimal skin with containing an image representing the appearance of the full skin. It also has the skin identifier that associates the preview with the corresponding skin.
- **Skin XML** – The skin markup language used by the Action Grid client software to define the resource and layout information, and render the skin pages.
- **Soft key** – A key on the device assigned to software shortcuts. For some devices, the user manuals refer to these keys as *selection keys*.
- **Status indicators** – Dynamic user interface elements provided by Action Grid for displaying current battery status, signal strength, appointments, and several others.