Workarounds & API Guide

Appeon® PowerServer® 2017

FOR WINDOWS & UNIX & LINUX

DOCUMENT ID: ADC20236-01-0700-01

LAST REVISED: October 27, 2017

Appeon Limited, 1/F, Shell Industrial Building, 12 Lee Chung Street, Chai Wan District, Hong Kong.

Contents

1 About This Book

1.1 Audience

This book is for developers who want to use APIs to implement various functionalities in the Web or mobile applications and get solutions for issues encountered when using PowerServer Web or PowerServer Mobile.

1.2 How to use this book

There are five chapters in this book.

Chapter 1: About This Book

A general description of this book.

Chapter 2: Appeon Workarounds PBL Reference

Lists the syntax and code examples of the programming interfaces provided by the Appeon Workarounds PBL, to implement corresponding functionalities.

Chapter 3: Appeon Resize PBL Reference

Lists the syntax and code examples of the programming interfaces provided by the Appeon Resize PBL, to resize and move the controls within a window.

Chapter 4: Workarounds for Unsupported Features

Provides suggestions to work around the unsupported features that have functional impact on the running of the application.

Chapter 5: FAQ and Workarounds

Lists some frequently asked questions and workaround tips regarding the Appeon application architecture or product features.

1.3 Related documents

Appeon provides the following user documents to assist you in understanding Appeon PowerServer and its capabilities:

- Introduction to Appeon:

  Gives general introduction to Appeon PowerServer and its editions.

- Getting Started (for PowerServer Mobile):

  Guides you though installing PowerBuilder and Appeon PowerServer, and developing and deploying a mobile application.

- New Features Guide:

  Introduces new features and changes in Appeon PowerServer.

- PowerServer Mobile Tutorials:

  Gives instructions on deploying, running, and debugging the mobile application, distributing native mobile apps, and configuring the PowerServer cluster.

- PowerServer Mobile (Offline) Tutorials:

  Gives instructions on setting up the PowerServer Mobile (Offline) environment, and configuring, deploying, running, updating, and debugging the offline application.

- Appeon Installation Guide:

  Provides instructions on how to install Appeon PowerServer successfully.

- Mobile UI Design & Development Guide:

  Introduces general guidelines on designing and developing the mobile app and UI.

- Migration Guidelines for PowerServer Web:

  A process-oriented guide that illustrates the complete diagram of the Appeon Web migration procedure and various topics related to steps in the procedure, and includes a tutorial that walks you through the entire process of deploying a small PowerBuilder application to the Web.

- Supported PB Features:

  Provides a detailed list of supported PowerBuilder features which can be converted to the Web/Mobile with Appeon as well as which features are unsupported.

- PowerServer Toolkit User Guide:

  Provides instructions on how to use the PowerServer Toolkit in Appeon PowerServer.

- Workarounds & API Guide:

  Provides resolutions for unsupported features and various APIs to facilitate you to implement the features (including Web and mobile) that are not easy or impossible to implement in the PowerBuilder IDE.

- Appeon Workspace User Guide:

  Gives a general introduction on Appeon Workspace and provides detailed instructions on how to use it.

- PowerServer Configuration Guide:

  Provides instructions on how to configure PowerServer Monitor, establish connections between PowerServer and database servers, and configure AEM for maintaining PowerServer and the deployed applications.

- Web Server Configuration Guide:

  Describes configuration instructions for different types of Web servers to work with the PowerServer.

- Troubleshooting Guide:

  Provides information on troubleshooting issues; covering topics, such as product installation, application deployment, AEM, and Appeon application runtime issues.

- Appeon Performance Tuning Guide:

Provides instructions on how to modify a PowerBuilder application to achieve better performance from its corresponding Web/mobile application.

- Testing Appeon Web Applications with UFT:

  Provides instructions on how to test Appeon Web applications with UFT.

1.4 If you need help

If you have any questions about this product or need assistance during the installation process, access the Technical Support Web site at http://www.appeon.com/support.

2 Appeon Workarounds PBL Reference

This chapter lists the syntax and code examples of the programming interfaces provided by the Appeon Workarounds PBL, to implement the various functionalities in the Appeon Web application and the Appeon mobile application.

2.1 Introduction to Appeon Workarounds

Appeon Workarounds provides a bunch of APIs and packaged solutions to facilitate you implement the features that are not easy or impossible to implement in the PowerBuilder IDE. Appeon Workarounds includes one PBL, three DLLs and one XML file to help to work around some PowerBuilder features and extend the usability of the deployed application.

- One PBL: appeon_workarounds.pbl
- One DLL: Eonejbclient.dll
- XML file: ejb_err_info.xml

They are located in the *developTempFile\appeon_workarounds* folder under the PowerServer installation directory. For example, *C:\inetpub\wwwroot\appeon\developTempFile \appeon_workarounds*.

To use the APIs or solutions provided in the Appeon Workarounds PBL, you will need to add the appeon_workarounds.pbl library to the Library Search Path of the application.

2.2 Best practices for calling APIs

Not all of the APIs can be used in both the Mobile and Web environment. In Section 2.3, “General API”, the following functions: of_popmenu, of_popmenuon, of_print2file, of_Print2PDF, of_getbrowserversion, of_getiehandle, of_getieurl, & of_is64browser from the AppeonExtFuncs object are effective in the Web environment only. In Section 2.4, “Mobile Device API”, all listed functions are effective in the Mobile environment. If a function that is effective in the Web environment only is executed in a Mobile environment, or if a function that is effective in the Mobile environment only is executed in a Web environment, the function may return unexpected values, and cause the application to perform abnormally.

Therefore, to avoid the aforementioned problem, you should detect the running environment first before calling the corresponding functions.

For example, to call the **of_setapprotationlock** API which take effects in the mobile environment only, you are recommended to write the scripts in this way:

```
if appeongetclienttype()="MOBILE" then
   eon_mobile_awsex lgnv_aws
   lgnv_aws = CREATE eon_mobile_awsex
   lgnv_aws.of_setAppOrientation(2)
   lgnv_aws.of_setapprotationlock(1)
    destroy lgnv_aws
end if
```

2.3 General API

The APIs listed in this section take effects in both the Web and mobile environment, except for those specified otherwise.

2.3.1 AppeonExtFuncs Object

AppeonExtfuncs object provides functions to help users manipulate the deployed application. The function of the AppeonExtfuncs object usually returns different values in the PowerBuilder application and in the deployed application. The values returned in the PowerBuilder application are defined in the AppeonExtfuncs object; while the values returned in the deployed application are defined in the Appeon client library.

The function of AppeonExtfuncs object performs almost identical to the global function of the Appeon client functions. Typically, the function of AppeonExtfuncs object is recommended over the Appeon client functions for the following two reasons:

- The function of the AppeonExtfuncs object delivers a better extensibility and is much easier to manage.
- Calling too much global functions of Appeon client functions may affect the performance.

Calling functions of AppeonExtFuncs object

The best way to use these functions is to pass their return values into PowerServer NVO components. Then, in the PowerServer NVO components, the information (such as browser version, user name etc.) can be utilized to code more application features such as security authentication, auditing, logging, file operation, etc. This means you can write more scripts in NVO components for implementing more application features.

Below lists the function of AppeonExtfuncs object and their corresponding Appeon client functions.

Table 2.1: Functions of AppeonExtFuncs object

Function of AppeonExtFuncs object	Appeon client functions	Description
of_getappeonusername function	AppeonGetAppeonUserName function	Gets the user name that you type into the Appeon Login dialog box.
of_getbrowserversion function	AppeonGetBrowserVersion function	Gets the Internet Explorer version of the client.
of_getcachedir function	AppeonGetCacheDir function	Gets the Cache directory that is used by the current application.
of_getclientid function	AppeonGetClientID function	Gets the unique session identifier for the Web client.
of_getclientip function	AppeonGetClientIP function	Gets the IP address of the Web client.
of_getclienttype function	AppeonGetClientType function	Gets the type of an application.
of_gethttpinfo function	AppeonGetHttpInfo function	Gets the HTTP header information from a particular request.
of_getiehandle function	AppeonGetIEHandle function	Gets the Internet Explorer handle for the current application.

Function of AppeonExtFuncs object	Appeon client functions	Description
of_getieurl function	AppeonGetIEURL function	Gets the URL of the application.
of_getostype function	AppeonGetOSType function	Gets the type of OS that runs your application (the Appeon Web application, the Appeon mobile application, or the PowerBuilder client application).
NA	AppeonGetRemainingdays function	Gets the remaining day(s) of license or technical support.
of_getservertype function	AppeonGetServerType function	Gets the PowerServer type where the application runs.
of_getsessioncount function (Obsolete)	AppeonGetSessionCount function (Obsolete)	This function is obsolete since Appeon PowerServer 2013. Replace it with the getSessionCount function in PowerServer open interfaces.
of_is64browser function	appeonisin64browser function	Detects if the IE browser where the application runs is 64-bit.
of_ldaplogon function	AppeonLDAPLogon Function	Logs in to the LDAP server with the specified user name and password.
of_popmenu function	AppeonPopMenu function	Pops up Appeon DataWindow menu at a specified position in a specified DataWindow control.
of_popmenuon function	AppeonPopMenuOn function	Pops up Appeon DataWindow menu in a specified window when you right click the mouse button.
of_print2file function	AppeonPrint2File function	Saves the specified DataWindow as image files of BMP, JPG or GIF format.
of_Print2PDF function (Obsolete)	AppeonPrint2PDF function (Obsolete)	of_Print2PDF is an obsolete function and will be discontinued in a future release. Please replace it with the SaveAs function of the DataWindow or Child DataWindow and set the saveas file type to PDF format.
NA	AppeonSetHintText function	Sets the HintText property of the SingleLineEdit control. The HintText property is not available in PowerBuilder SingleLineEdit control. It is specially designed for the Appeon mobile application.
of_switchRealTimeCal function	AppeonSwitchRealTimeCal function	Performs the DataWindow real-time expression calculation in time or performs the calculation for only one time in the whole life-cycle.

2.3.1.1 of_clearlog function

Description

Clears the Web/mobile client log information.

This function is supported on both the Web and mobile clients.

Syntax

appeonextfuncs.of_clearlog ( )

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.

Return value

integer

1 - Success.

-1 - It is called in PowerBuilder, or there is an error.

2.3.1.2 of_getappeonadtversion function

Description

Gets the version number of PowerServer Toolkit.

Syntax

appeonextfuncs.of_getappeonadtversion (string *as_version*)

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.
as_version	The version number of PowerServer Toolkit.

Return value

String.

Usage

This function takes effect in the deployed Appeon application, not in the original PowerBuilder application.

2.3.1.3 of_getappeonusername function

Description

Gets the user name that you type into the Appeon Login dialog box.

Syntax

appeonextfuncs.of_getappeonusername ( )

Argument	Description

appeonextfuncs	A reference to an AppeonExtFuncs object.

Return value

String.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.1.4 of_getappeonwebversion function

Description

Gets the version number of Appeon Web Library.

Supported on the Web client only.

Syntax

appeonextfuncs.of_getappeonwebversion (string *as_version*)

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.
attribute	The version number of Appeon Web Library or an empty string if it is called in the mobile client or if there is any error.

Return value

Integer.

1 - Success.

-1 - It is called in the mobile client, or there is an error.

Usage

This function takes effect in the deployed Appeon application, not in the original PowerBuilder application.

2.3.1.5 of_getbrowserversion function

Description

Gets the Internet Explorer version of the Web client.

Supported in the browser-based Web application only.

Syntax

appeonextfuncs.of_getbrowserversion ( )

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.

Return value

String.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.1.6 of_getcachedir function

Description

Gets the cache directory that is used by the current application.

For the Web application, the cache directory can be customized in AEM. For details, refer to Section 5.4.6.7, "Client Storage Location" in *PowerServer Configuration Guide for .NET* or in PowerServer Configuration Guide for J2EE.

For the iOS mobile application, the cache directory is /$Appeon_Mobile_folder$/Documents/$your_application_folder$, for example, */var/mobile/Applications/144F5F33-A33F-480D-A3D9-01BBA5410EB2/Documents/4c001b05*.

For the Android mobile application, the cache directory is /data/data/$Appeon_Mobile_folder$/files/$your_application_folder$, for example, */data/data/com.appeon.mobile/files/21746870*.

Syntax

```
appeonextfuncs.of_getcachedir()
```

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.

Return value

String.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.1.7 of_getclientid function

Description

Gets the unique session identifier for the Web or mobile client.

Syntax

```
appeonextfuncs.of_getclientid()
```

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.

Return value

String.

2.3.1.8 of_getclientip function

Description

Gets the IP address of the Web or mobile client.

Syntax

appeonextfuncs.of_getclientip ( )

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.

Return value

String.

2.3.1.9 of_getclienttype function

Description

Gets the type of an application.

Syntax

appeonextfuncs.of_getclienttype ( )

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.

Return value

String.	Returns "WEB" if the application runs in the Web browser or on the desktop as an IWA (installable Web app). Returns "PB" if the application runs in PowerBuilder. Returns "MOBILE" if the application runs on a mobile device.

2.3.1.10 of_gethttpinfo function

Description

Gets the HTTP header information from a particular request.

Syntax

appeonextfuncs.of_gethttpinfo (string *attribute*)

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.

attribute	The required HTTP information. For example, "Host", "Cookie", etc.

Return value

String.

Usage

This function takes effect in the deployed Appeon application, not in the original PowerBuilder application.

2.3.1.11 of_getiehandle function

Description

Gets the Internet Explorer handle for the Web application.

Supported in the browser-based Web application only.

Syntax

appeonextfuncs.of_getiehandle()

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.

Return value

Long.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.1.12 of_getieurl function

Description

Gets the URL of the Web application.

For the mobile application, use of_getappinfo to get the URL of the mobile application.

Syntax

appeonextfuncs.of_getieurl()

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.

Return value

String.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.1.13 of_getostype function

Description

Gets the type of OS that runs your application (the Appeon Web application, the Appeon mobile application, or the PowerBuilder client application).

Syntax

appeonextfuncs.of_getostype()

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.

Return value

String.	Returns the type of OS that runs the Appeon Web application, the Appeon mobile application, or the PowerBuilder client application.

2.3.1.14 of_getrunmode function

Description

Gets the running mode of an application.

Syntax

appeonextfunc.of_getrunmode()

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.

Return value

String	Returns "" an empty string if the application runs in PowerBuilder. Returns "RELEASE" or "DEBUG" (Debug JS, or debug JS/PS) (according to the generation mode setting in PowerServer Toolkit, see Section 4.2.1.2.1, “Basic Settings” in *PowerServer Toolkit User Guide*) if the application runs on the Web or a mobile device.

2.3.1.15 of_getservertype function

Description

Gets the PowerServer type where the application runs.

Syntax

appeonextfuncs.of_getservertype()

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.

Return value

Integer	Returns 1 if the Appeon applications runs on a PowerServer that is installed to a Java Server (such as JBoss). Returns 2 if the Appeon applications runs on a PowerServer that is installed to a .NET IIS server.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.1.16 of_getsessioncount function

This function is obsolete since Appeon PowerServer 2013. Replace it with the getSessionCount function in PowerServer open interfaces.

2.3.1.17 of_is64browser function

Description

Detects if the IE browser where the Web application runs is 64-bit.

Supported in the browser-based Web application only.

Syntax

```
appeonextfuncs.of_is64browser()
```

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.

Return value

Boolean	Returns "true" if the Appeon application runs on a 64-bit IE browser. Returns "false" if the Appeon application runs on a 32-bit IE browser.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.1.18 of_ldaplogon function

Description

Logs in to the LDAP server with the specified user name and password.

Syntax

appeonextfuncs.of_ldaplogon (string *as_username*, string *as_password*)

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.
as_username	User name for logging into the LDAP server.
as_password	Password for logging into the LDAP server.

Return value

String.	Returns "" an empty string if the login succeeds. Returns the error information.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application. To make this function work in the Appeon application, you should set the System Security to ON in AEM, and make sure the LDAP server is correctly set up. For detailed instructions, refer to Section 5.3.5.5, “System Security” in *PowerServer Configuration Guide for .NET* or PowerServer Configuration Guide for J2EE.

2.3.1.19 of_log function

Description

Generates the Web/mobile client log information, and the default log level is INFO.

This function is supported on both the Web and mobile clients.

Syntax

appeonextfuncs.of_log (integer *ai_level*, string *as_info*)

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.
ai_level	1 - FATAL level. 2 - ERROR level. 3 - WARNING level. 4 - INFO level. (Default value) 5 - DEBUG level.
as_info	The log information.

Return value

integer

1 - Success.

-1 - It is called in PowerBuilder, or there is an error.

Usage

For the log path of Web applications, mobile applications (that run in Appeon Workspace), and native mobile applications, please refer to Section 5.4.6.8, “Client Logs” in *PowerServer Configuration Guide for .NET* or PowerServer Configuration Guide for J2EE.

2.3.1.20 of_popmenu function

Description

Pops up Appeon DataWindow menu at a specified position in a specified DataWindow control.

Appeon DataWindow Menu is available in PowerServer Web only. For more information about the menu items, refer to Section 8.4, “Appeon DataWindow menu” in *PowerServer Toolkit User Guide*.

Supported on the Web client only.

Syntax

appeonextfuncs.of_popmenu (datawindow *adw_dw* , integer *nx* , Integer *ny*)

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.
adw_dw	The DataWindow control on which you want to pop up the Appeon DataWindow menu.
nx	The distance from the left edge of the popup menu.
ny	The distance from the top edge of the popup menu.

Return value

None.

Usage

1. The user customized RMB menu has a higher priority than the Appeon customized menu.

2. The AppeonPopMenu function has a higher priority than the AppeonPopMenuOn function.

3. Defining your RMB menu in RButtonDown event is not recommended because this will cause confusion in the system. To work around this, define your RMB menu in the RButtonUp event.

2.3.1.21 of_popmenuon function

Description

Pops up Appeon DataWindow menu in a specified window when you right click the mouse button.

Appeon DataWindow Menu is available in PowerServer Web only. For more information about the menu items, refer to Section 8.4, “Appeon DataWindow menu” in *PowerServer Toolkit User Guide*.

Supported on the Web client only.

Syntax

appeonextfuncs.of_popmenuon (datawindow *adw_dw*, Boolean *ab_show*)

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.
adw_dw	The DataWindow control on which you want to show the Appeon DataWindow menu.
ab_show	Gives an option whether to display the Appeon DataWindow menu. True - Display the Appeon DataWindow menu. False - Not to display the Appeon DataWindow menu.

Return value

None.

Usage

1. The user customized RMB menu has a higher priority than the Appeon customized menu.
2. The AppeonPopMenu function has a higher priority than the AppeonPopMenuOn function.
3. Defining your RMB menu in RButtonDown event is not recommended because this will cause the confusion of the system. To work around this, define your RMB menu in the RButtonUp event.

2.3.1.22 of_print2file function

Description

Saves the specified DataWindow as image files of BMP, JPG or GIF format.

Supported on the Web client only.

Syntax

appeonextfuncs.of_print2file (datawindow *adw*, string *asoutpath*, string *asoutname*, long *alouttype*)

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.
adw	The DataWindow object to be saved as image files.

asoutpath	The path of image files.
asoutname	The specified name of image files.
alouttype	The format type of files: 1-BMP; 2-JPG; 3-GIF

Return value

Integer	Returns 1 if it succeeds in saving the specified DataWindow as image files. Returns -1 if an unknown error occurs. Returns -2 if alouttype is an unsupported format. Returns -3 if adw is an invalid DataWindow object, DataStore object or DataWindowChild object. Returns -4 if it fails in creating a file, e.g., the specified path does not exist or without access rights. Returns -5 if it fails in creating device context, e.g., user sets a large size when customizing page property. Returns -6~-12 if an internal error occurs.

Usage

1. This function is used to execute saving DataWindow as image files.

2. If the page size is large enough, a DataWindow is saved as one file; if the size of a DataWindow surpasses the page size, the DataWindow is saved as several files.

3. The asoutname is the file name specified by user, for example, the function is appeonprint2file(adw, "c:\", "appeon", 1), if a DataWindow is saved as one file, the file is like C:\appeon.bmp; if a DataWindow is saved as several files, the files are like C:\appeon1.bmp, C:\appeon2.bmp, ..., C:\appeonN.bmp. Another example, the function is appeonprint2file(adw, "c:\", "test.bmp", 1), if a DataWindow is saved as one file, the file is like C:\ test.bmp.bmp; if a DataWindow is saved as several files, the files are like C:\test.bmp1.bmp, C:\test.bmp2.bmp, ..., C:\test.bmpN.bmp.

4. The upper limit of page size is restrained by the type of operating system. For instance, in Windows Vista, the upper limit of customized page size is 5500*5500 around. However, the page size can also be 1024*10000 by reducing page width and increasing page height.

2.3.1.23 of_Print2PDF function (Obsolete)

Obsolete function

of_Print2PDF is an obsolete function and will be discontinued in a future release. Please replace it with the SaveAs function of the DataWindow or Child DataWindow and set the saveas file type to PDF format. For details, refer to Section 11.10.3, “Functions of DataWindow control” in *Supported PB Features for PowerServer Web* or Supported PB Features for PowerServer Mobile.

Supported on the Web client only.

2.3.1.24 of_switchRealTimeCalc function

Description

Performs the DataWindow real-time expression calculation in time or performs the calculation for only one time in the whole life-cycle.

Syntax

appeonextfuncs.of_switchRealTimeCalc (powerobject *adw*, integer *para*)

Argument	Description
appeonextfuncs	A reference to an AppeonExtFuncs object.
adw	The DataWindow/DataStore/DataWindowChild object.
para	0 - Default value, performs the real-time calculation; 1 - No need to perform the real-time calculation, and performs the calculation for only one time in the whole life-cycle.

Return value

Integer.	0 - Success. -1 - Failure.

2.3.2 Appeon Client Functions

Appeon client functions are PowerBuilder global functions which perform identical functionalities to the function of AppeonExtFuncs object. However, heavy use of global functions are not recommended due to their poor extensibility and performance compared to object functions, therefore, whenever it is feasible, Appeon client function should be replaced with the equivalent function of AppeonExtFuncs object.

2.3.2.1 AppeonGetAppeonUserName function

Description

Gets the user name that you type into the Appeon Login dialog box.

Syntax

AppeonGetAppeonUserName ( )

Return value

String.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.2.2 AppeonGetBrowserVersion function

Description

Gets the Internet Explorer version of the Web client.

Supported in the browser-based Web application only.

Syntax

```
AppeonGetBrowserVersion()
```

Return value

String.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.2.3 AppeonGetCacheDir function

Description

Gets the cache directory that is used by the current application.

For the Web application, the cache directory can be customized in AEM. For details, refer to Section 5.4.6.7, “Client Storage Location” in *PowerServer Configuration Guide for .NET* or in PowerServer Configuration Guide for J2EE.

For the iOS mobile application, the cache directory is /$Appeon_Mobile_folder$/Documents/$your_application_folder$, for example, */var/mobile/Applications/144F5F33-A33F-480D-A3D9-01BBA5410EB2/Documents/4c001b05*.

For the Android mobile application, the cache directory is /data/data/$Appeon_Mobile_folder$/files/$your_application_folder$, for example, */data/data/com.appeon.mobile/files/21746870*.

Syntax

```
AppeonGetCacheDir()
```

Return value

String.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.2.4 AppeonGetClientID function

Description

Gets the unique session identifier for the Web or mobile client.

Syntax

```
AppeonGetClientID()
```

Return value

String.

2.3.2.5 AppeonGetClientIP function

Description

Gets the IP address of the Web or mobile client.

Syntax

```
AppeonGetClientIP()
```

Return value

String.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.2.6 AppeonGetClientType function

Description

Gets the type of an application.

Syntax

```
AppeonGetClientType()
```

Return value

String	Returns "WEB" if the application runs in the Web browser or on the desktop as an IWA (installable Web app). Returns "PB" if the application runs in PowerBuilder. Returns "MOBILE" if the application runs on a mobile device.

2.3.2.7 AppeonGetClientVersion function

Description

Gets the version of the Appeon client that runs your application.

Syntax

```
AppeonGetClientVersion()
```

Return value

String.

2.3.2.8 AppeonGetHttpInfo function

Description

Gets the HTTP header information from a particular request.

Syntax

`AppeonGetHttpInfo` (string *`attribute`*)

Argument	Description
`attribute`	The required HTTP information. For example, "Host", "Cookie", etc.

Return value

String

2.3.2.9 AppeonGetIEHandle function

Description

Gets the Internet Explorer handle for the Web application.

Supported in the browser-based Web application only.

Syntax

`AppeonGetIEHandle` ( )

Return value

Long.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.2.10 AppeonGetIEURL function

Description

Gets the URL of the Web application.

For the mobile application, use of_getappinfo to get the URL of the mobile application.

Syntax

`AppeonGetIEURL` ( )

Return value

String.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.2.11 AppeonGetOSType function

Description

Gets the type of OS that runs your application (the Appeon Web application, the Appeon mobile application, or the PowerBuilder client application).

Syntax

`AppeonGetOSType` ( )

Return value

String.	Returns the type of OS that runs the Appeon Web application, the Appeon mobile application, or the PowerBuilder client application.

2.3.2.12 AppeonGetRemainingdays function

Description

Gets the remaining day(s) of license or technical support.

Syntax

`AppeonGetRemainingdays` (String *`as_type`*, ref string *`as_error`*)

Argument	Description
`as_type`	License or technical support that you want to get the remaining day(s). "license" indicates to get remaining day(s) of license. "support" indicates to get remaining day(s) of technical support.
`as_error`	An empty string or error messages.

Return value

Long.	Returns a number >0 if the license or technical support has remaining day(s). Returns 0 if there is no expiration date. Returns -1 if license or technical support has expired, PowerServer has an exception, or parameter is invalid.

2.3.2.13 AppeonGetServerType function

Description

Gets the PowerServer type where the application runs.

Syntax

`AppeonGetServerType` ( )

Return value

Integer.	Returns 1 if the application runs on a PowerServer that is installed to a Java Server (such as JBoss). Returns 2 if the application runs on a PowerServer that is installed to a .NET IIS server.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.2.14 AppeonGetSessionCount function

This function is obsolete since Appeon PowerServer 2013. Replace it with the getSessionCount function in PowerServer open interfaces.

2.3.2.15 appeonisin64browser function

Description

Detects if the IE browser where the application runs is 64-bit.

Syntax

```
appeonisin64browser()
```

Return value

Boolean.	Returns "true" if the application runs on a 64-bit IE browser. Returns "false" if the application runs on a 32-bit IE browser.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application.

2.3.2.16 AppeonLDAPLogon function

Description

Logs in to the LDAP server with the specified user name and password.

Syntax

AppeonLDAPLogon (string *as_username*, string *as_password*)

Argument	Description
as_username	User name for logging into the LDAP server.
as_password	Password for logging into the LDAP server.

Return value

String.

Usage

This function takes effect in the deployed Appeon application, not in the PowerBuilder application. To make this function work in the deployed application, you should set the System Security to **ON** in AEM, and make sure the **LDAP server** is correctly set up. For detailed instructions, refer to Section 5.3.5.5, “System Security” in *PowerServer Configuration Guide for .NET* or PowerServer Configuration Guide for J2EE.

2.3.2.17 AppeonPopMenu function

Description

Pops up Appeon DataWindow menu at a specified position in a specified DataWindow control.

Appeon DataWindow Menu is available in PowerServer Web only. For more information about the menu items, refer to Section 8.4, “Appeon DataWindow menu” in *PowerServer Toolkit User Guide*.

Supported on the Web client only.

Syntax

AppeonPopMenu (datawindow *adw_dw* , Integer *nx* , Integer *ny*)

Argument	Description
adw_dw	The DataWindow control on which you want to pop up the Appeon DataWindow menu.
nx	The instance from the left edge of the popup menu.
ny	The instance from the top edge of the popup menu.

Return value

None.

Usage

1. The user customized RMB menu has a higher priority than the Appeon customized menu.
2. The AppeonPopMenu function has a higher priority than the AppeonPopMenuOn function.
3. Defining your RMB menu in RButtonDown event is not recommended because this will cause confusion in the system. To work around this, define your RMB menu in the RButtonUp event.

2.3.2.18 AppeonPopMenuOn function

Description

Pops up Appeon DataWindow menu in a specified window when you right click the mouse button.

Appeon DataWindow Menu is available in PowerServer Web only. For more information about the menu items, refer to Section 8.4, “Appeon DataWindow menu” in *PowerServer Toolkit User Guide*.

Supported on the Web client only.

Syntax

AppeonPopMenuOn (datawindow *adw_dw*, Boolean *ab_show*)

Argument	Description
adw_dw	The DataWindow control on which you want to show the Appeon DataWindow menu.

Argument	Description
ab_show	Gives an option whether to display or not display the Appeon DataWindow menu. True - Enables the display of the Appeon DataWindow menu. False - Disables the display of the Appeon DataWindow menu.

Return value

None.

Usage

1. The user customized RMB menu has a higher priority than the Appeon customized menu.
2. The AppeonPopMenu function has a higher priority than the AppeonPopMenuOn function.
3. Defining your RMB menu in RButtonDown event is not recommended because this will cause the confusion of the system. To work around this, define your RMB menu in the RButtonUp event.

2.3.2.19 AppeonPrint2File function

Description

Saves the specified DataWindow as image files of BMP, JPG or GIF format.

Supported on the Web client only.

Syntax

`AppeonPrint2File` (datawindow *`adw`*, string *`asoutpath`*, string *`asoutname`*, long *`alouttype`*)

Argument	Description
`adw`	The DataWindow object to be saved as image files
`asoutpath`	The path of image files
`asoutname`	The specified name of image files
`alouttype`	The format type of files: 1-BMP; 2-JPG; 3-GIF

Return value

Integer.	Returns 1 if it succeeds in saving the specified DataWindow as image files. Returns -1 if an unknown error occurs.

	Returns -2 if alouttype is an unsupported image format. Returns -3 if adw is an invalid DataWindow object, DataStore object or DataWindowChild object. Returns -4 if it fails in creating a file, e.g., the specified path does not exist or without access rights. Returns -5 if it fails in creating device context, e.g., user sets a large size when customizing page property. Returns -6~-12 if an internal error occurs.

Usage

1. This function is used to execute saving DataWindow as image files.

2. If the page size is large enough, a DataWindow is saved as one file; if the size of a DataWindow surpasses the page size, the DataWindow is saved as several files.

3. The asoutname is the file name specified by user, for example, the function is appeonprint2file(adw, "c:\", "appeon", 1), if a DataWindow is saved as one file, the file is like C:\appeon.bmp; if a DataWindow is saved as several files, the files are like C:\appeon1.bmp, C:\appeon2.bmp, ..., C:\appeonN.bmp. Another example, the function is appeonprint2file(adw, "c:\", "test.bmp", 1), if a DataWindow is saved as one file, the file is like C:\ test.bmp.bmp; if a DataWindow is saved as several files, the files are like C:\test.bmp1.bmp, C:\test.bmp2.bmp, ..., C:\test.bmpN.bmp.

4. The upper limit of page size is restrained by the type of operating system. For instance, in Windows Vista, the upper limit of customized page size is 5500*5500 around. However, the page size can also be 1024*10000 by reducing page width and increasing page height.

2.3.2.20 AppeonPrint2PDF function (Obsolete)

Obsolete function

AppeonPrint2PDF is an obsolete function and will be discontinued in a future release. Please replace it with the SaveAs function of the DataWindow or Child DataWindow and set the saveas file type to PDF format. For details, refer to Section 11.10.3, “Functions of DataWindow control” in *Supported PB Features for PowerServer Web* or Supported PB Features for PowerServer Mobile.

Supported on the Web client only.

2.3.2.21 AppeonSetHintText function

Description

Sets the HintText property of the SingleLineEdit control. The HintText property is not available in PowerBuilder SingleLineEdit control. It is specially designed for the Appeon mobile application.

Note: This function will be obsolete in the next release after 2017.

`appeonsethinttext` (singlelineedit *`sle_target`*, string *`as_hinttext`*)

Argument	Description
sle_target	The SingleLineEdit control.
as_hinttext	The text that will be displayed as hint text in the SingleLineEdit control.

Return value

Integer.	1 - Success. -1 - It is called in PowerBuilder or PowerServer Web, or there is an error. -2 - The *sle_target* is invalid. -3 - The *as_hinttext* argument is null.

2.3.2.22 AppeonSetLongPressInterval function

Description

Sets the duration (in milliseconds) for a press before it is recognized as a long press. Once this duration is reached, the keyboard will be brought up to display. By default, the keyboard pops up when the user taps twice in the field of the DataWindow, or when the user presses the DataWindow field for as long as 1500 milliseconds.

Supported on mobile client only.

Syntax

`appeonsetlongpressinterval` (long *al_interval*)

Argument	Description
al_interval	Sets the duration in milliseconds. If it is set to a value smaller than 500 milliseconds, then 550 milliseconds will be used, because 500 milliseconds is already used as the duration for the DataWindow drag & drop event.

Return value

Long.	1 - Success. -1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

2.3.2.23 AppeonSwitchRealTimeCalc function

Description

Performs the DataWindow real-time expression calculation in time or performs the calculation for only 1 time in the whole life-cycle.

Syntax

`AppeonSwitchRealTimeCalc` (powerobject *adw*, integer *para*)

Argument	Description
adw	The DataWindow/DataStore/ DataWindowChild object.
para	0 - Default value, performs the real-time calculation; 1 - No need to perform the real-time calculation, and performs the calculation for only one time in the whole life-cycle.

Return value

Integer.	0 - Success. -1 - Failure.

2.3.3 Client Logs

2.3.3.1 eon_logex object

Description

Facilitates users to debug the deployed applications.

The of_log function and of_clearlog function of eon_logex object call the corresponding function of AppeonExtFuncs object respectively, please refer to of_log function and of_clearlog function of AppeonExtFuncs object for details.

These two functions are supported on both the Web and mobile clients.

2.3.4 File Upload and Download

Appeon provides a non-visual object, AppeonFileService object, with five functions for uploading files to and downloading files from the file server. This is a web-based solution, and it does not work in your client/server application.

You should follow the steps below to implement File Upload and Download:

Step 1: Configuring and deploying the file server

Step 2: Uploading and Downloading files

2.3.4.1 Configuring and deploying Appeon File Server

Appeon File Server is a standard non-visual Web application running on the back-end for uploading and downloading files. For PowerServer installed to the .NET IIS, the setup wizard of Appeon File Server (setup.exe) can be found under *%IIS_Web_Root%\appeon\plugin \fileservice*. For PowerServer installed to the Java server, the WAR package of Appeon File Server (fileservice.war) can be found under *%Appeon_Server%\plugin\fileservice*.

You must configure and deploy the file server first in order to use the AppeonFileService object in the Appeon Workarounds PBL. In most cases Appeon File Server is deployed to the machine where the application server is installed. It can also be deployed separately. For

example, when the file size is too big, uploading or downloading it will negatively impact the performance of the application server, you can deploy the file server separately to another machine.

In the following section, you will be guided to configure and deploy Appeon File Server to IIS (.NET Framework), WebLogic, WebSphere, JBoss, JEUS, and NetWeaver. For more about the deploy instructions, refer to related documents of IIS, WebLogic, WebSphere, JBoss, JEUS, and NetWeaver.

2.3.4.1.1 Deploying Appeon File Server to IIS

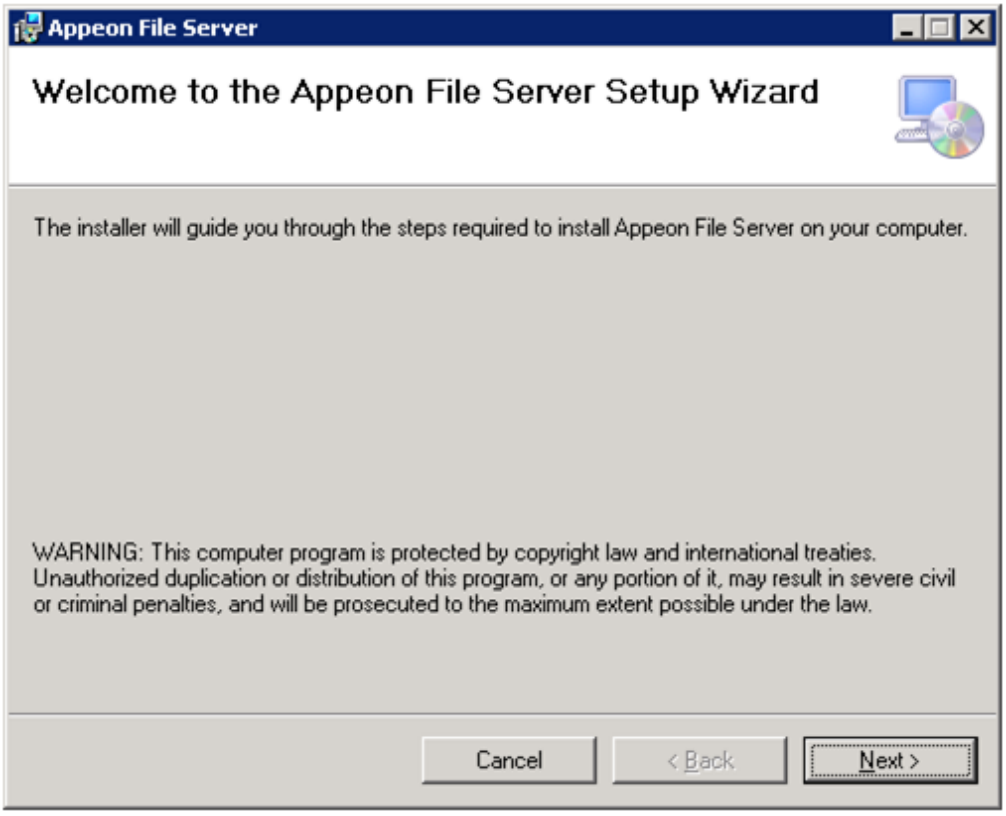
Installing the Appeon File Server

You will need to install first and then configure the Appeon File Server. For installing the file server to IIS, Appeon provides a **setup.exe** file at *%IIS_Web_Root%\appeon\plugin \fileservice*.

Step 1: Double click the **setup.exe** file to start the setup.

If the "Installation Incomplete" error displays, please try the solution in the Section 1.4.1, “Appeon File Server Installation Incomplete” in *PowerServer Troubleshooting Guide*.

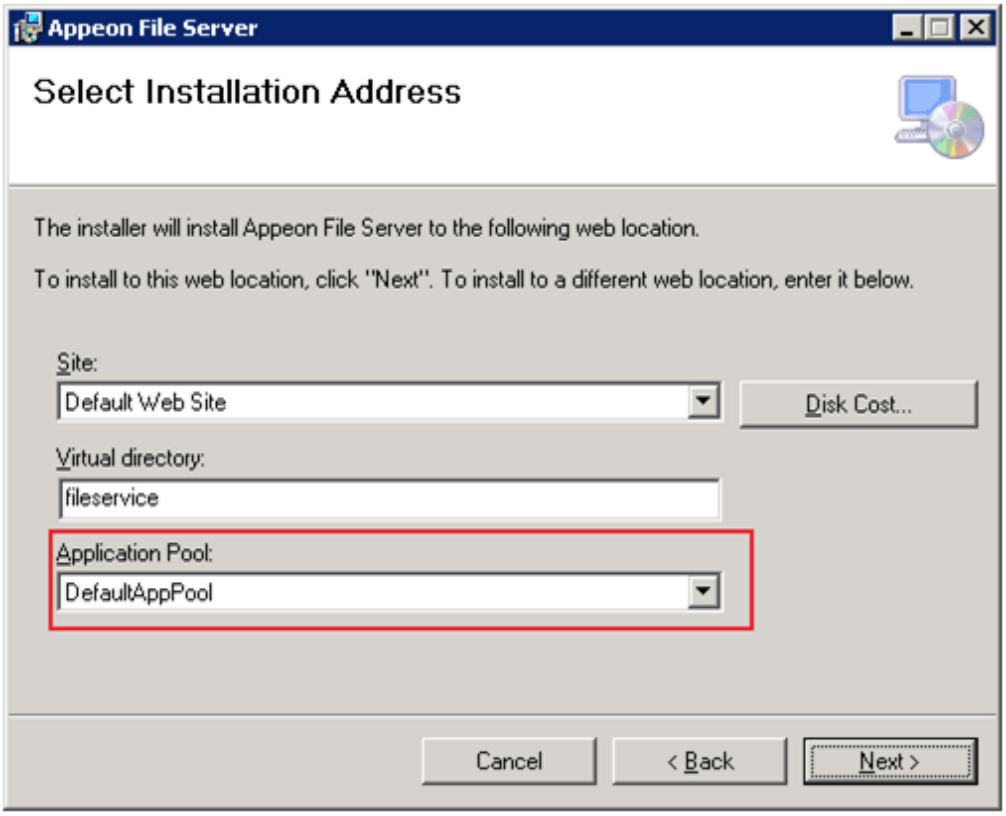
Figure 2.1: Appeon File Service Setup Wizard



Step 2: Click the **Next** button and then the following window pops up. Select a Web site where the Appeon File Server will be installed.

Note that DO NOT change the name of Virtual directory otherwise Appeon File Server will fail to start.

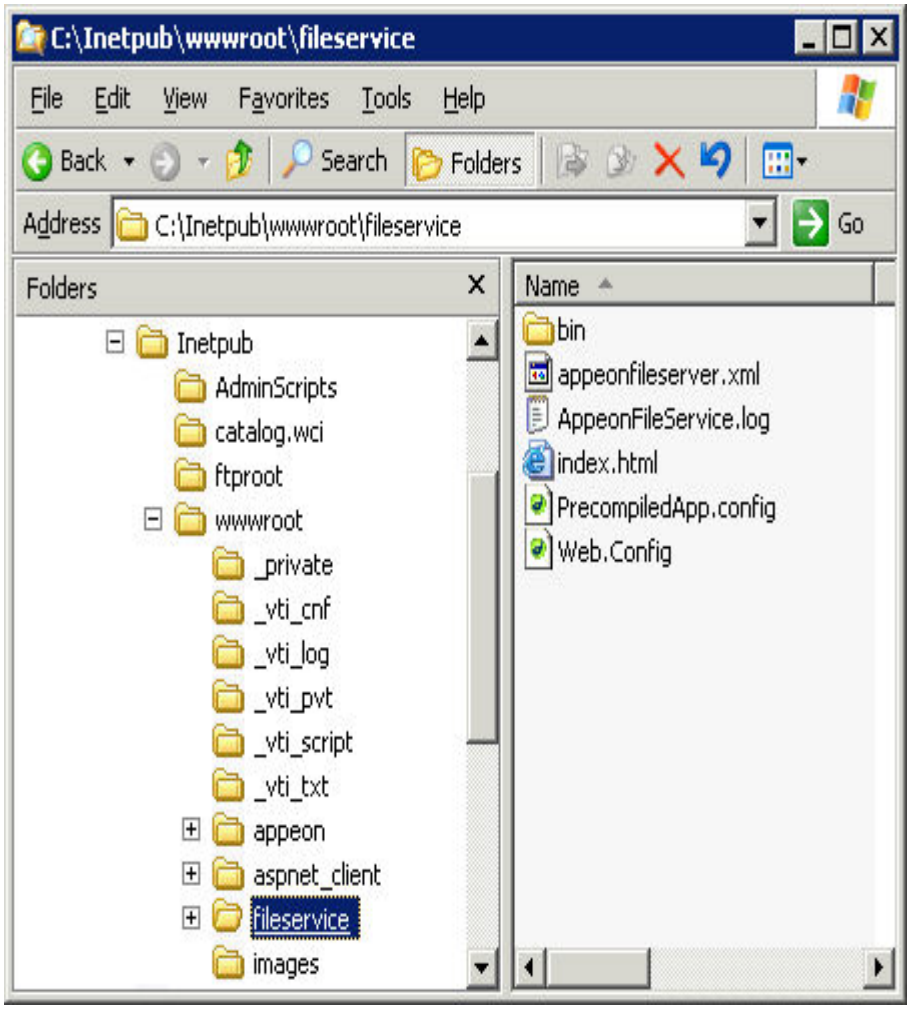
Figure 2.2: Select Installation Address



Step 3: Click **Next** until the installation is complete.

Step 4: Go to directory where the Appeon File Server is installed, for example (*C:\inetpub \wwwroot\fileservice*).

Figure 2.3: Appeon File Server directory

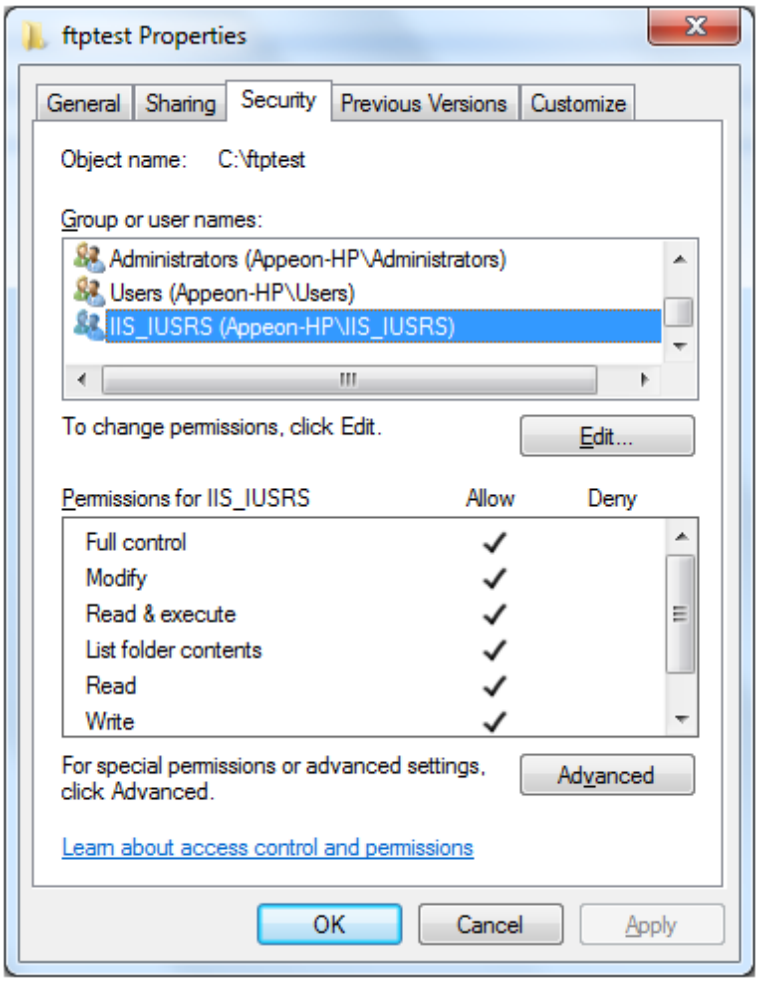


Step 5: Select **Properties** from the right-click menu of the **fileservice** folder.

Step 6: In the **Security** tab page of the **fileservice Properties** window,

For **IIS 7.5 or above**, grant the **IIS_IUSRS** user with **Full Control** permission to this folder.

Figure 2.4: Grant Full Control (IIS7)



Uninstall Appeon File Server

Step 1: Go to **Start** | **Control Panel** | **Add and Remove Programs**.

Step 2: Select **Appeon File Server** and click **Remove** button.

Configuring the Appeon File Server

After deploying the Appeon File Server, follow steps below to configure it.

Open the appeonfileserver.xml in *C:\Inetpub\wwwroot\fileservice*.

```
<?xml version="1.0" encoding="UTF-8"?>
<webserver>
<!-- The value of attribute "value" must begin with either a single or double quote
 character. -->
 <file-path value="D:\appeon\upload" />
 <log-level value="3" />          <!--0 Error,1 Info,2 Func, 3 Debug-->
 <session-timeout value="3600" /> <!-- the unit is second  -->
 <allowed-file-types value="txt;doc;jpg;mpeg" ignorecase="true" />
 <max-file-size value="20" />     <!-- the unit is M -->
 <users>
  <user name="test" password="password" />
  <user name="userA" password="userA" />
  <user name="userB" password="userB" />
 </users>
```

```
</webserver>
```

Modify the file according to your own demands.

Table 2.2: Settings specification value

Settings	The value of the setting specifies...
<file-path>	Specifies the directory to which files are saved after uploaded to the file server. Files are automatically saved in the "plugin" folder under the application directory when downloaded from the file server.
<log-level>	Specify the log level. Value: 0 - Error 1 - Information 2 - Function 3 - Debug
<session-timeout>	Specifies the time, in seconds, that passes after the last request is processed before the session times out.
<allow-file-type>	Specifies the file types that can be uploaded. Use a ";" between two file types. 1. "*" - Any file types are allowed. Note: Do not use "*" with other file types because "*" will not take any effect. For example, "*;text" means that only text file is allowed. 2. Using a "-" before the file types - Any file type is allowed, excluding the listed ones. For example, "-text;doc;jpg;mpeg" means that any file type is allowed, excluding text, doc, jpg and mpeg files. 3. "Ignorecase" - The file type is not case sensitive.
<max-file-size>	Specifies the maximum size (MB) of the file that can be uploaded.
<user>	Specifies the User Name and Password that can log on the file server. Multiple users are allowed as shown in the code example above.

2.3.4.1.2 Deploying Appeon File Server to WebLogic

Configuring the Appeon File Server

You will need to configure first and then deploy the Appeon File Server. Follow steps below to configure the Appeon File Server:

Step 1: On the machine where the Appeon File Server will be deployed, create an XML file wherever you like and name it whatever you wish. In this example, create an XML file named "**appeonfileserver.xml**" under *D:\appeon\config directory*.

Step 2: Copy the following code to the XML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<fileserver>
```

```
 <file-path value="D:\appeon" />
 <session-timeout value="3600" />
 <allowed-file-types value="txt;doc;jpg;mpeg" ignorecase="true" />
 <max-file-size value="20" />
 <users>
  <user name="userA" password="userA" />
  <user name="userB" password="userB" />
 </users>
</fileserver>
```

Step 3: Modify the settings in the XML file if necessary.

Table 2.3: The settings specification value

Settings	The value of the setting specifies...
<file-path>	Specifies the directory to which files are saved after uploaded to the file server. Files are automatically saved in the "plugin" folder under the application directory when downloaded from the file server.
<session-timeout>	Specifies the time, in seconds, that passes after the last request is processed before the session times out.
<allow-file-type>	Specifies the file types that can be uploaded. Use a ";" between two file types. 1. "*" - Any file types are allowed. Note: Do not use "*" with other file types because "*" will not take any effect. For example, "*;text" means that only text file is allowed. 2. "-" - Exclude the file types listed after "-". For example, "-text;doc;jpg;mpeg" means that any file type is allowed, except for text, doc, jpg and mpeg files. 3. "Ignorecase" - The file type is not case sensitive.
<max-file-size>	Specifies the maximum size (MB) of the file that can be uploaded.
<user>	Specifies the User Name and Password that can log on the file server. Multiple users are allowed as shown in the code example above.

Step 4: On the machine where the PowerServer is installed, unzip the **fileservice.war** (*%powerserver%\plugin\fileservice*) and find the **web.xml** file under the **WEB-INF** folder. Then open **web.xml** with a text editor. Replace the **bold** text with the name and directory of the XML file created in the previous steps.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC '-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN'
'http://java.sun.com/dtd/web-app_2_3.dtd'>
<web-app>

<servlet>
 <servlet-name>fileservice</servlet-name>
 <servlet-class>com.appeon.fileserver.WebServiceDispatcher</servlet-class>
 <init-param>
  <param-name>config</param-name>
  <param-value>D:\appeon\config\appeonfileserver.xml</param-value>
 </init-param>
```

```
 <load-on-startup>1</load-on-startup>
</servlet>
<servlet>
 <servlet-name>uploadfile</servlet-name>
 <servlet-class>com.appeon.fileserver.UploadFile</servlet-class>
 <init-param>
  <param-name>config</param-name>
  <param-value>D:\appeon\config\appeonfileserver.xml</param-value>
 </init-param>
 <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
 <servlet-name>fileservice</servlet-name>
 <url-pattern>/fileservice</url-pattern>
</servlet-mapping>

</web-app>
```

Step 5: Save the file and re-compress the **fileservice** folder to **fileservice.war** using WinZip, WinRAR or JDK. Do not use the other compression tools such as 7-zip.

Then follow the next section Deploying the Appeon File Server to deploy fileservice.war.

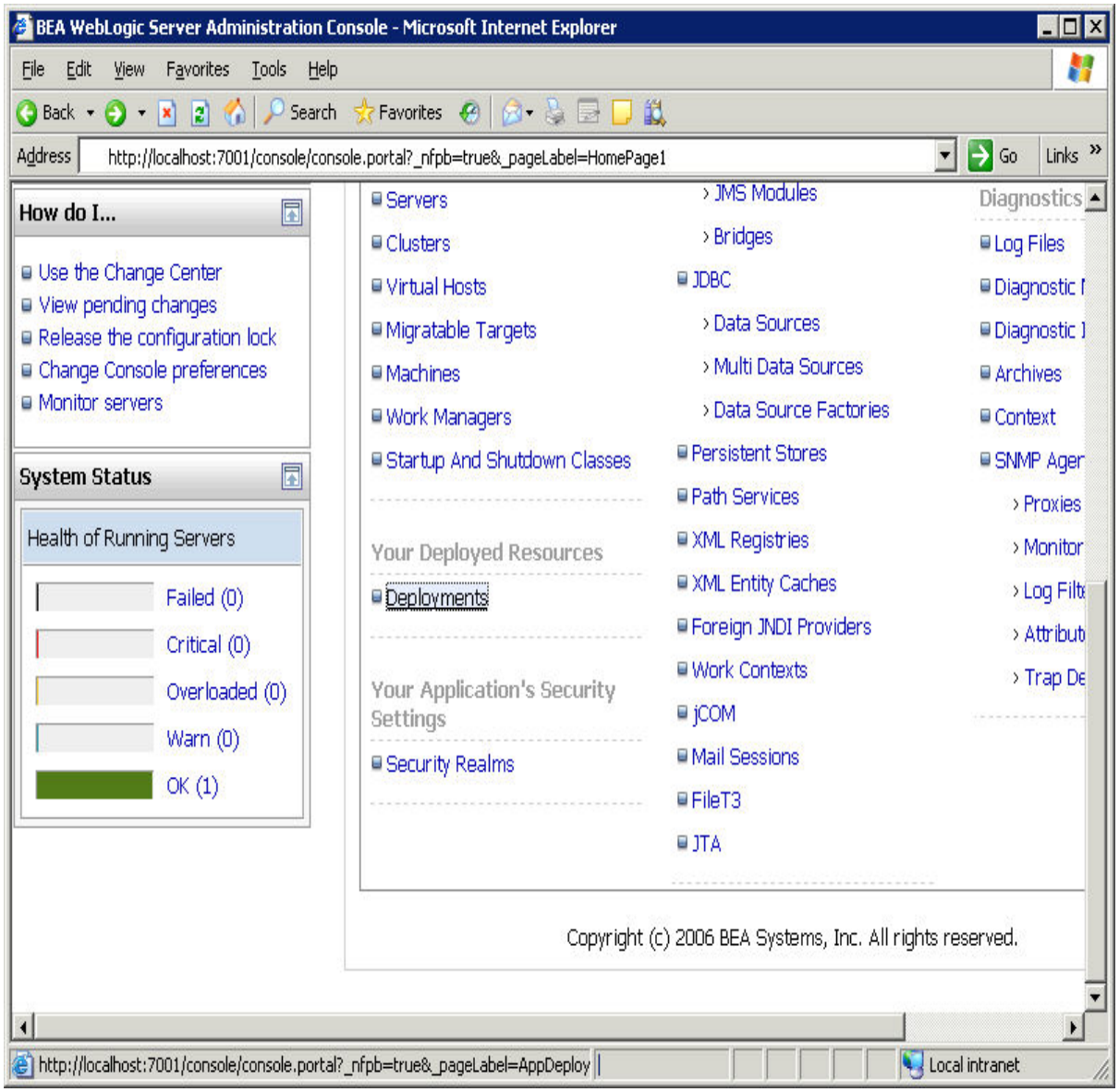
Deploying the Appeon File Server

After configuring the Appeon File Server, follow steps below to deploy it:

Step 1: On the machine where PowerServer is installed, access the **WebLogic Server Administration Console** in a Web browser.

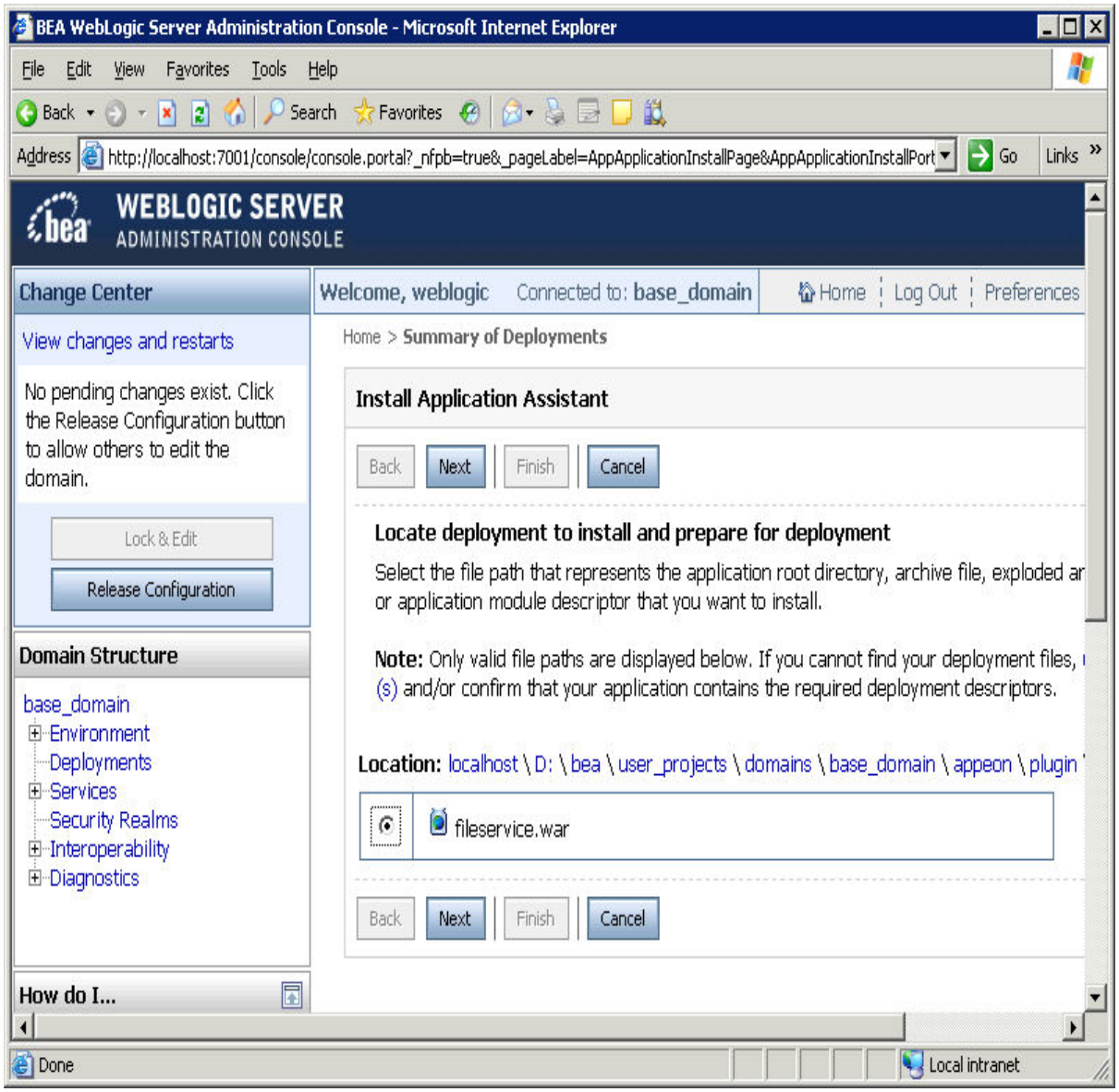
Step 2: Click the **Deployment** link.

Figure 2.5:



Step 3: Click the **Install** button and select the **fileservice.war** file at *%powerserver%\plugin \fileservice*.

Figure 2.6:



Step 4: Click the **Next** button to use the default settings and then click the **Finish** button to finish the deployment of Appeon File Server.

2.3.4.1.3 Deploying Appeon File Server to WebSphere

Configuring the Appeon File Server

Configuring the Appeon File Server in the Java server such as WebLogic, WebSphere, JBoss, JEUS, NetWeaver etc. is the same. Please refer to Configuring the Appeon File Server for WebLogic.

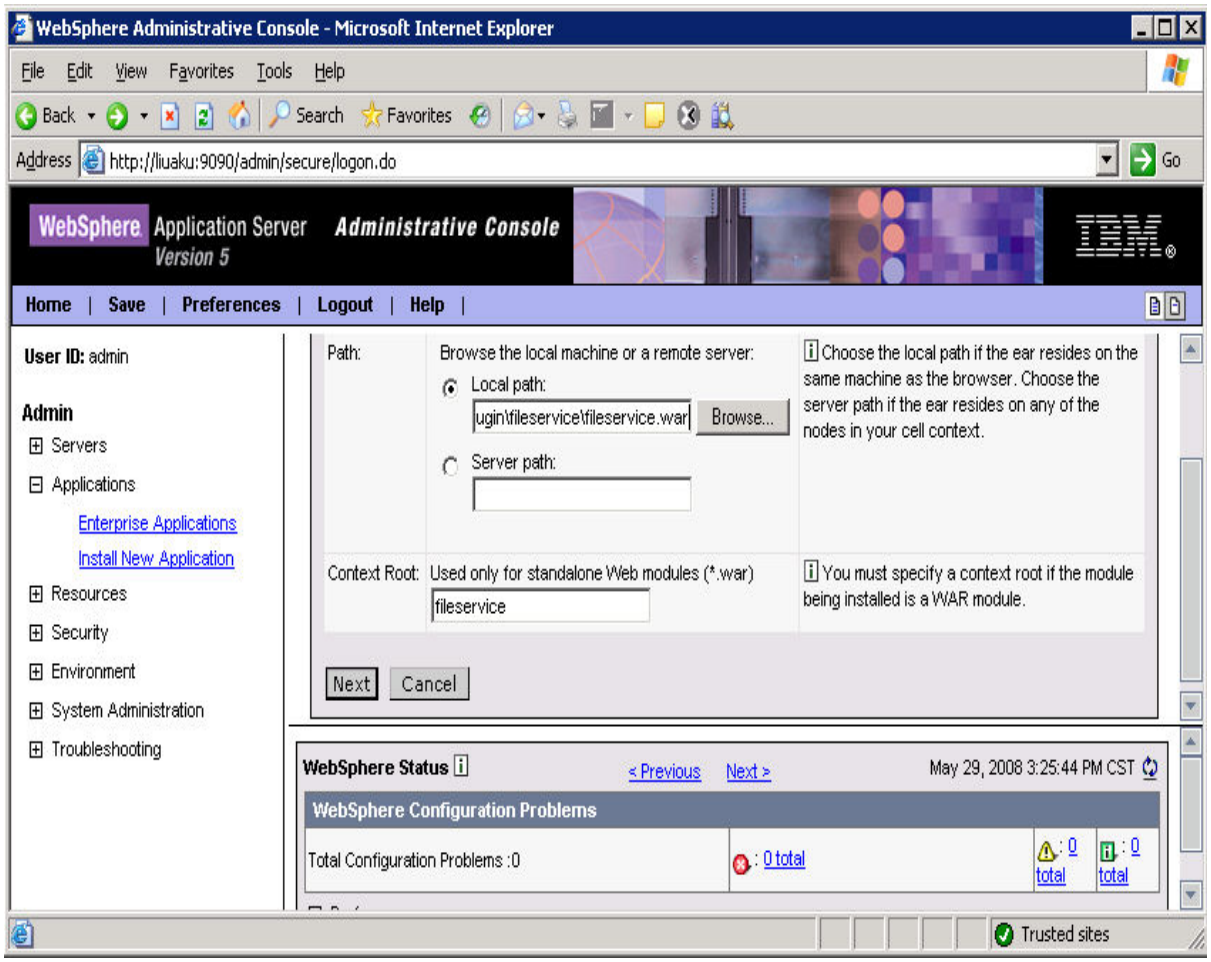
Deploying the Appeon File Server

After configuring the Appeon File Server, follow steps below to deploy it:

Step 1: On the machine where PowerServer is installed, access the **WebSphere Administrative Console** in a Web browser.

Step 2: Select **Applications** | **Install New Applications** from the left tree view. And then in the right page specify the path where the **fileservice.war** is (by default at *%powerserver% \plugin\fileservice*) and **Context Root** for the WAR file and click the **Next** button.

Figure 2.7:



Step 3: Click **Next** with default settings until the file server is successfully installed on the WebSphere server.

2.3.4.1.4 Deploying Appeon File Server to JBoss

Configuring the Appeon File Server

Configuring the Appeon File Server in the Java server such as WebLogic, WebSphere, JBoss, JEUS, NetWeaver etc. is the same. Please refer to Configuring the Appeon File Server for WebLogic.

Deploying the Appeon File Server

After configuring the Appeon File Server, follow steps below to deploy it. There are two methods to deploy the Appeon File Server in JBoss:

- Automatic deploy

  Copy the file **fileservice.war** to the directory %JBoss installation root directory% \standalone\deployments and then start JBoss.

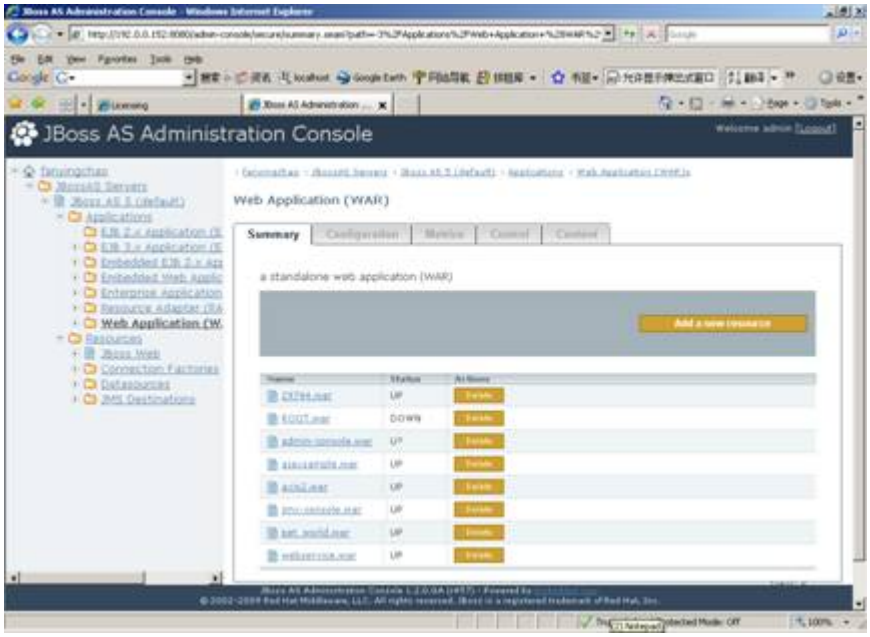
The fileservice.war file will be deployed automatically.

- Manual deploy

1. Start JBoss and log in to JBoss console.

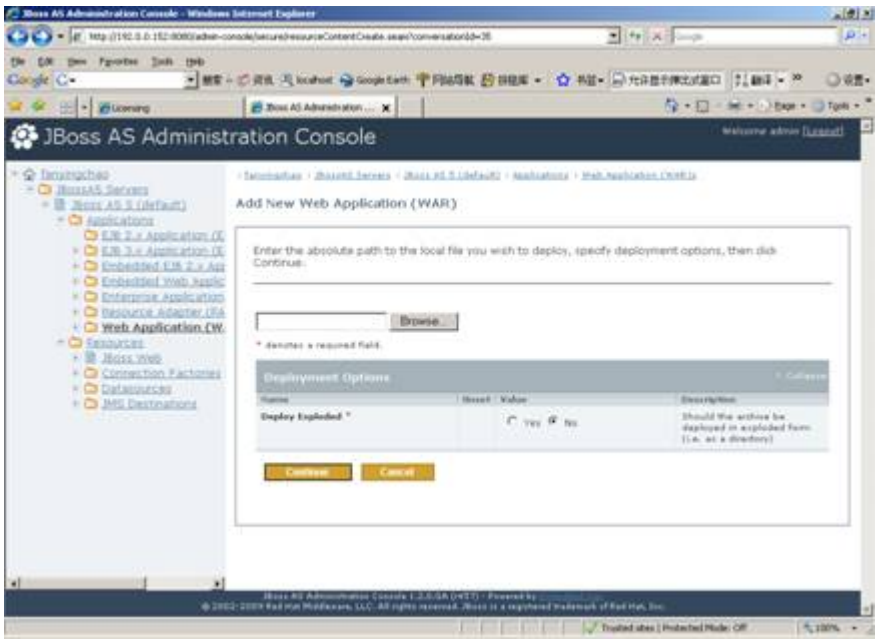
2. Select **Web Application** in the left tree view and then click **Add a new resource** in the right page.

Figure 2.8: PDF



3. Click **Browse** and locate the file **fileservice.war**.

Figure 2.9:



4. Click **Continue** with default settings until the file server is successfully deployed in the JBoss server.

2.3.4.1.5 Deploying Appeon File Server to JEUS

Configuring the Appeon File Server

Configuring the Appeon File Server in the Java server such as WebLogic, WebSphere, JBoss, JEUS, NetWeaver etc. is the same. Please refer to Configuring the Appeon File Server for WebLogic.

Deploying the Appeon File Server

After configuring the Appeon File Server, follow steps below to deploy it:

Copy the **fileservice.war** file to the directory *%JEUS installation root directory%\webhome\autodeploy* and then start JEUS server. The fileservice.war file will be deployed automatically.

2.3.4.1.6 Deploying Appeon File Server to NetWeaver

Configuring the Appeon File Server

Configuring the Appeon File Server in the Java server such as WebLogic, WebSphere, JBoss, JEUS, NetWeaver etc. is the same. Please refer to Configuring the Appeon File Server for WebLogic.

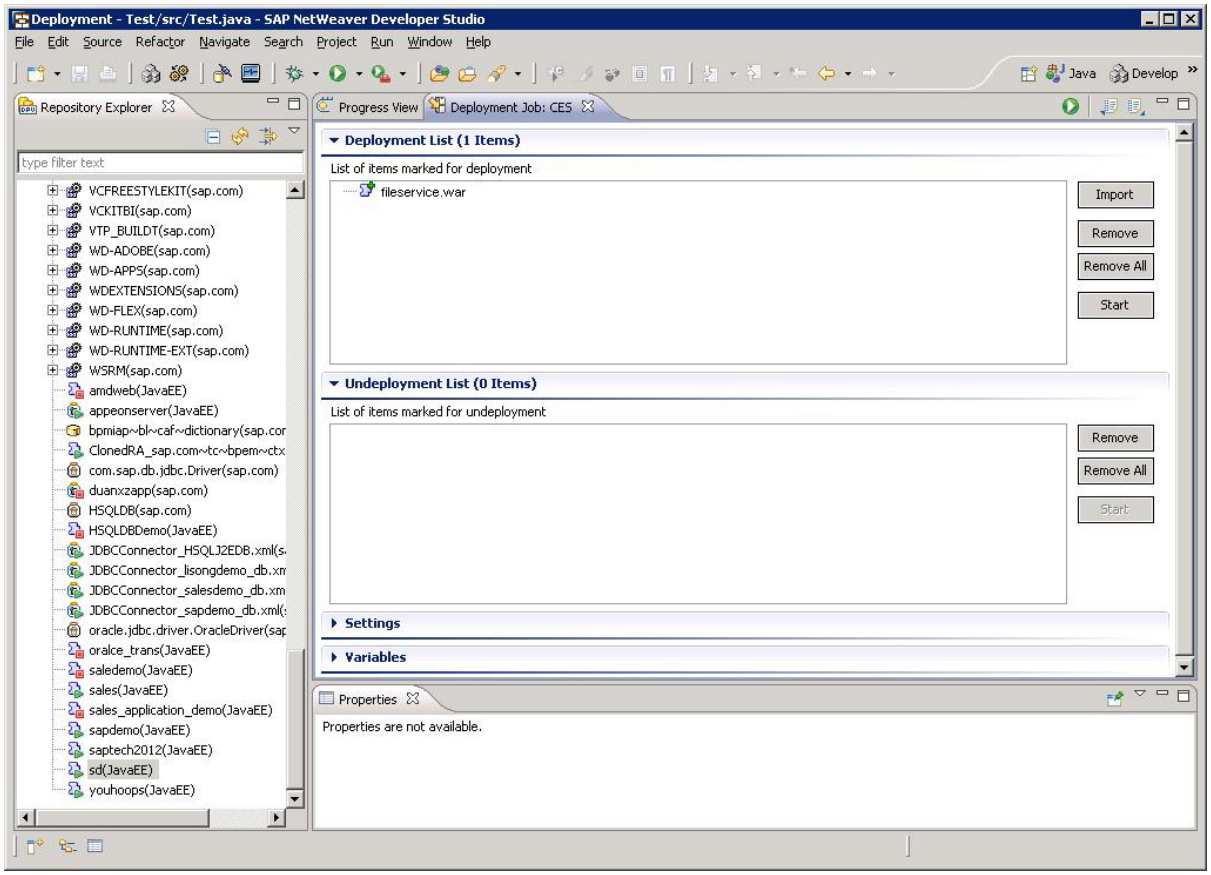
Deploying the Appeon File Server

After configuring the Appeon File Server, follow steps below to deploy it:

Step 1: Open the deployment perspective in the SAP NetWeaver Developer Studio by clicking the menu **Window** | **Open Perspective** | **Other** and then selecting **deployment** in the pop-up window. The deployment perspective is shown as below.

Step 2: Click the **Import** button to import the **fileservice.war** file and then click **Start** to deploy the Appeon File Server after the file is imported successfully.

Figure 2.10:



2.3.4.2 Uploading and downloading files

After configuring and deploying the Appeon File Server, you can follow the steps below to upload files to or download files from the Appeon File Server.

Step 1: Add Appeon Workaround PBL to your PowerBuilder project.

Step 2: Define a variable and create an instance for the **AppeonFileService** object.

Step 3: Call of_logonfileserver to connect to the file server.

Step 4: Call of_appeonupload or of_appeondownload to upload or download files.

Step 5: Call of_logofffileserver to disconnect from the file server.

2.3.4.2.1 AppeonFileService object

AppeonFileService object provides service about the file uploading and downloading.

Functions for AppeonFileService object

AppeonFileService object provides the following functions to upload and download files:

- of_logonfileserver function: Connects to the file server.
- of_logofffileserver function: Disconnects from the file server.
- of_appeonupload function: Uploads file to the file server. You can specify the path where the source file locates and the path where the file will be uploaded.

- of_appeondownload function: Downloads the specified file from the file server. You can specify the path from which the file is downloaded and the path to which the file will be saved.
- of_FileExists function: Reports whether the specified file exists on the file server.

of_logonfileserver

Description

Connects to the Appeon File Server.

Syntax

appeonfileservice.of_logonfileserver (string *fileserveripaddress*, long *port*, string *connectstring*)

Parameter	Description
appeonfileservice	An instance of an AppeonFileService object
fileserveripaddress	Protocol and IP address or machine name of the Appeon File Server. To use the HTTPS protocol, you should add "https://" before the IP address or machine name, for example, "https://192.168.168.52", otherwise, HTTP protocol (http://) will be used by default.
port	HTTP or HTTPS port of the application where the Appeon File Server is deployed.
connectstring	User name and password for connecting to the Appeon File Server. Format: "username=username;password=password". Use a semicolon ";" to separate the user name and password. The user name and password should be consistent with those specified in the appeonfileserver.xml configuration file.

Return value

Long.	Return values are: 0 - The logon is successful. -1 - Connection to the file server fails. -2 - The provided user name or password is invalid.

Code example

```
long ll_re
string info

ll_re = fileservice.of_logonfileserver ("192.168.168.52", 80,
 "username=userA;password=userA")
```

```
if ll_re = 0 then
 info = "Logon to file server is successful."
elseif  ll_re = 1 then
 info = "Failed to connect to the file server."
else
 info = "Failed to connect to the file server."
end if

Messagebox( "ll_re="+string(ll_re),info)
```

of_logofffileserver

Description

Disconnects from the Appeon File Server.

Syntax

appeonfileservice.of_logofffileserver()

Parameter	Description
appeonfileservice	An instance of an AppeonFileService object

Return value

Long.	Return values are: 0 - The log-off is successful. -1 - Connection to the Appeon File Server fails. -2 - The session has timed out or the user has not logged off the file server.

Code example

```
long ll_re
string info

ll_re = fileservice.of_logofffileserver ( )

if ll_re = 0 then
 info = "Logoff file server is successful."
elseif  ll_re = 1 then
 info = "Failed to log off file server."
else
 info = "The session has timed out or the user has not logged off the file server."
end if

Messagebox( "ll_re="+string(ll_re),info)
```

of_appeonupload

Description

Uploads file to the Appeon File Server.

Syntax

appeonfileservice.of_appeonupload (string *source*, string *destination*, boolean *isrename*, ref long *errorcode*)

appeonfileservice.of_appeonupload (string *destination*, boolean *isrename*, ref long *errorcode*) (Web only)

appeonfileservice.of_appeonupload (boolean *isrename*, ref long *errorcode*) (Web only)

Parameter	Description
appeonfileservice	An instance of an AppeonFileService object.
source	The file name and directory of the source file. For the mobile application, this parameter is **required**; it must be specified with a valid directory. For the Web application, this parameter is **optional**. If no directory is specified here, a dialog box will be displayed prompting the user to select a file to upload.
destination	The destination directory on the file server where the file is uploaded.The directory specified here is a relative path to the file directory specified in the appeonfileserver.xml configuration file. This parameter is **optional** to both the Web application and the mobile application.
isrename	Whether to rename the file if another file with the same name already exists.
errorcode	The error occurred when uploading files. Values are: 0 - Succeeded. -1 - Connection to the file server fails. -2 - ID error. -3 - The format of the source file is forbidden. -4 - Destination directory is invalid or the file is too large.

Return value

String.	Returns the URL of the uploaded file if it succeeds and returns an empty string if it fails. Returns NULL if any argument is NULL.

Usage

If you want to delete an uploaded file, you must restart the file server after deleting it. Otherwise, you will fail to upload the file with the same name.

Code example

```
long ll_re
string info,ls_url
```

```
ls_url = fileservice.of_appeonupload ("C:\temp\1.pdf", "pdf_files", true, ll_re)

if ll_re = 0 then
 info = "Upload is successful."
elseif  ll_re = 1 then
 info = "Failed to upload the file."
else
 info = "Other errors."
end if

Messagebox("ll_re="+string(ll_re),info+"~r~nurl="+ls_url)
```

of_appeondownload

Description

Downloads the specified file from the Appeon File Server.

Syntax

appeonfileservice.of_appeondownload (string *source*)

appeonfileservice.of_appeondownload (string *source*, string *target*)

Parameter	Description
appeonfileservice	An instance of an AppeonFileService object.
source	The file name and directory of the file which will be downloaded. The directory specified here is a relative path to the file directory specified in the appeonfileserver.xml configuration file.
target	(Optional) The destination directory on the client machine where the file will be saved after downloaded from the file server. For the mobile application, this parameter is ignored, as the file will be automatically saved to the "plugin" folder under the application directory. You can determine the directory for the current application by calling the AppeonGetCacheDir function. For the Web application, this parameter is optional. If no directory is specified here, a dialog box will be displayed prompting the user to select a location to save the file.

Return value

Long.	Return values are: 0 - The file is successfully downloaded. -1 - The file download failed or exceptions occurred.

	-3 - File name is empty. NULL if any argument is NULL.

Code example

The following code example is to download a PDF file named "1.pdf" from the "pdf_files" folder under the specified directory on the Appeon File Server. On the Web client, the file will be saved to "C:\" as specified by the *target* parameter; while on the mobile device, the file will be automatically saved to the "plugin" folder for the current application, despite that the *target* parameter is set to "C:\".

```
long ll_re
string info

ll_re = fileservice.of_appeondownload ("pdf_files\1.pdf", "C:\")

if ll_re = 0 then
 info = "Download is successful."
else
 info = "Other errors."
end if

Messagebox ("ll_re="+string(ll_re),info)
```

The following code example is to view the PDF file (named "1.pdf" located on the local C drive) via Internet Explorer. This is effective for the Web client only.

```
string ls_file

ls_file = "~"C:\Program Files\Internet Explorer\iexplore.exe~" C:\1.pdf"
Messagebox("",ls_file)
run(ls_file)
```

of_FileExists

Description

Reports whether the specified file exists on the Appeon File Server.

Syntax

appeonfileservice.of_FileExists (String *filename*)

Parameter	Description
appeonfileservice	An instance of the AppeonFileService object.
filename	A string whose value is the name of a file.

Return value

Long.

1 - The file exists.

0 - The file does not exist on the Appeon File Server.

-1 - Failed to connect to the Appeon File Server.

Code example

```
long ll_re
string info

ll_re = fileservice.of_fileexists ("pdf_files\1.pdf")

if ll_re =1 then
 info = "The file exists."
elseif ll_re =0 then
 info = "The file does not exist on the file server."
else
 info = "Other errors."
end if

Messagebox( "ll_re="+string(ll_re),info)
```

of_downloadfile (Obsolete)

Obsolete function

of_downloadfile is an obsolete function and will be discontinued in a future release. Please replace it with of_appeondownload.

of_uploadfile (Obsolete)

Obsolete function

of_uploadfile is an obsolete function and will be discontinued in a future release. Please replace it with of_appeonupload.

Properties for AppeonFileService object

AppeonFileService object provides the following properties (private instances) to upload and download files:

- is_FileServerName instance: Indicates the name of the file server.
- is_LogOnParams instance: Indicates the connection string for logging into the file server.
- il_Id instance: Long. Indicates the ID number returned from the file server when logon has succeeded.

is_FileServerName

Description

Indicates the IP address or machine name of the file server. This property is initialized after the of_LogOnFileServer function is called.

Return value

String.

is_LogOnParams

Description

Indicates the connection string for logging into the file server. This property is initialized after the of_LogOnFileServer function is called.

Return value

String.

il_Id

Description

Indicates the ID number returned from the file server when logon has succeeded. This property is initialized after the of_LogOnFileServer function is called. It is used by the file server to validate the client.

Return value

Long.

2.3.5 Appeon Labels

Overview

Appeon Labels provided in the Appeon Workarounds PBL can reduce the interactions between the client and PowerServer, thus boosting the performance of Appeon applications.

The Appeon Labels associated functions are contained in the appeon_nvo_db_update object in appeon_workarounds.pbl.

Appeon Labels and associated functions

Label Name	Label Associated Function	Description
Appeon Commit/ Rollback Label	of_autocommitrollback	Automatically commits or rolls back the first database operation statement after the label.
Appeon Commit Label	of_autocommit	Automatically commits the first database operation.
Appeon Rollback Label	of_autorollback	Automatically rolls back the first database operation statement if the operation fails.
Appeon Queue Labels	of_startqueue of_commitqueue	The Appeon Queue Labels are designed for use when it is necessary to execute numerous database operation statements on PowerServer, and the returned values of the statements are not validated or used. Using the pair of labels can dramatically reduce the number of client-server interactions.
Appeon Immediate Call Label	of_imdcall	Immediately commits a database operation.
Appeon Update Label	of_update	Reduces the number of interactions with the server caused by "interrelated updates".

Usage

For detailed instructions on how to take advantages of Appeon Labels to improve performance of the application, please refer to the Section 6.5, “Technique #4: grouping multiple server calls with Appeon Labels” in *PowerServer Performance Tuning Guide*.

2.3.5.1 Appeon Commit/Rollback Label

Description

The Appeon Commit/Rollback Label is used to automatically commit or rollback the first database operation statement after the label.

Controls

appeon_nvo_db_update object

Associated functions

of_autocommitrollback

Syntax

objectname.of_autocommitrollback( )

Argument	Description
objectname	An instance of the appeon_nvo_db_update object.

Return value

None.

Usage

With the Appeon Commit/Rollback Label, the database operation statement will be sent to the PowerServer. The server will automatically commit (or roll back) the statement according to the execution result. If the execution succeeds, the result will be committed; if the execution fails, the result will be rolled back.

The first Commit or Rollback statement after the Appeon Commit/Rollback Label will not be submitted to the PowerServer. Therefore, there must be no more than one database operation statement between the label and the first Commit or Rollback statement. For example, the IF statement should not contain database operation statements, since the executed result will not be committed to the database.

```
gnv_appeonDbLabel.of_autocommitrollback()
UPDATE tab_a ......
IF SQLCA.SQLCODE = 0 THEN
   ...... // non-database related bussiness logic
   COMMIT;
ELSE
   ...... // non-database related bussiness logic
   ROLLBACK;
   ......
END IF
```

There must be database related operations after the label.

There must be no labels between the Appeon Rollback Label and the first Commit or Rollback statement.

2.3.5.2 Appeon Commit Label

Description

The Appeon Commit Label is used to automatically commit the first database operation.

Controls

appeon_nvo_db_update object

Associated functions

of_autocommit

Syntax

objectname.of_autocommit( )

Argument	Description
objectname	An instance of the appeon_nvo_db_update object.

Return value

none.

Usage

After the label there must be database operations.

With Appeon Commit Label, PowerServer does not validate the execution result of the database operation statement. Instead, the server automatically commits the statement regardless of the execution result.

The first Commit statement after the Appeon Commit Label will not be submitted to the PowerServer , however, the first Rollback statement will be submitted to the server. Therefore, there should be no more than one database operation between the label and the first Commit statement. For example, the IF statement should not contain database related business logic, since the executed result will not be committed to the database.

```
gnv_appeonDbLabel.of_autocommit()
SELECT.....INTO ......FROM tab_a;
IF SQLCA.SQLCODE = 0 THEN
  ...... // non-database related business logic
ELSE
  ...... // non-database related business logic
END IF
COMMIT;
```

There must be no labels between the Appeon Commit Label and the first Commit statement.

2.3.5.3 Appeon Rollback Label

Description

The Appeon Rollback Label is used to automatically roll back the first database operation statement if the operation fails.

Controls

appeon_nvo_db_update object

Associated functions

of_autorollback

Syntax

objectname.of_autorollback( )

Argument	Description
objectname	An instance of the appeon_nvo_db_update object.

Return value

None.

Usage

After the label there must be database operations.

With the Appeon Rollback Label, PowerServer only commits an unsuccessful database operation.

The first Rollback statement after the Appeon Rollback Label will not be submitted to the PowerServer if the execution fails. Therefore, there should be no more than one database operation between the label and the first Commit or Rollback statement.

There must be no labels between the Appeon Rollback Label and the first Rollback statement.

Code Example

```
gnv_appeonDbLabel.of_autorollback()
IF dw_1.update() <> 1 THEN
  ROLLBACK ;
  ...... // non-database related business logic
END IF
```

2.3.5.4 Appeon Queue Labels

Description

Appeon Queue Labels consist of the Appeon Start Queue Label and the Appeon Commit Queue Label. The Appeon Queue Labels are designed for use when it is necessary to execute numerous database operation statements on PowerServer, and the returned values of the statements are not validated or used. Using the pair of labels can dramatically reduce the number of client-server interactions.

Controls

appeon_nvo_db_update object

Associated functions

of_startqueue, of_commitqueue

Syntax

objectname.of_startqueue ( )

objectname.of_startqueue({integer *stopmode*})

Argument	Description
objectname	An instance of the appeon_nvo_db_update object.
stopmode	0 - continue executing the remaining SQL scripts when an error occurs; 1 - stop executing the remaining SQL scripts when an error occurs. Note: The of_startqueue function without this argument is preserved for compatibility.

objectname.of_commitqueue()

Argument	Description
objectname	An instance of the appeon_nvo_db_update object.

Return value

none.

Usage

Appeon Queue Labels must be used in the same field.

All the database operations in the labels will be submitted to the PowerServer.

If there are multiple Appeon Commit Queue Labels used together with an Appeon Start Queue Label, only the first Appeon Commit Queue Label that is executed will be effective. Other Appeon Commit Queue Labels will be ignored.

With the stopmode argument, users can choose to continue running or return immediately when an error occurs in the database syntax operation in the queue.

In the Appeon Queue Labels, the SELECT statement cannot be used in the condition statements. The following example is incorrect.

Incorrect Example

```
nv_appeonDbLabel.Of_startqueue()
IF.....THEN
  SELECT STATEMENT 1
else
  SELECT STATEMENT 2
END IF
gnv_appeonDbLabel.Of_commitqueue()
```

In the Appeon Queue Labels, script that stops the execution of another script cannot be included in some events of the DataWindow object, For example, in the following events of DataWindow, the Return statement should not be used: the RetrieveStart event, the RetrieveEnded event, the RowFocusChanged event, the UpdateStart event, the UpdateEnd event, and etc.

For every RETURN statement, there must be an database operation statement or unexpected errors occur.

You can open a cursor in the Appeon Queue Labels.

Using multiple Appeon Queue Labels

Appeon Queue Labels can be embedded in other Appeon Queue Labels. However, only the outer Appeon Queue Labels take effect.

Using non-queue labels together with Appeon Queue Labels

When there are multiple non-queue labels embedded in the Appeon Queue Labels, only the first non-queue label takes effect.

When the other Appeon Labels is embedded in Appeon Queue Labels, the format should be the same as the following code example. Please note that only Commit or Rollback statements are involved in the condition statements.

Code Example

```
nv_appeonDbLabel.of_startqueue()
dw_1.update()
gnv_appeonDbLabel.of_autocommitrollback() // the label takes effect
gnv_appeonDbLabel.of_imdcall() // The label takes no effect
IF dw_2.update() = 1 THEN
  COMMIT;
ELSE
  ROLLBACK;
END if

nv_appeonDbLabel.of_startqueue(1) // Stop immediately when an error occurs
dw_1.update()
gnv_appeonDbLabel.of_autocommitrollback() // the label takes effect
gnv_appeonDbLabel.of_imdcall() // The label takes no effect
if dw_2.update() = 1 THEN
  COMMIT;
ELSE
  ROLLBACK;
END if

nv_appeonDbLabel.of_autocommitrollback()
UPDATE tab_a......
if SQLCA.SQLCODE = 0 THEN
  COMMIT;
ELSE
  ROLLBACK;
END IF
INSERT tab_b......
  COMMIT;
gnv_appeonDbLabel.of_commitqueue()
```

2.3.5.5 Appeon Immediate Call Label

Description

The Appeon Immediate Call Label is used to immediately commit a database operation.

Controls

appeon_nvo_db_update object

Associated functions

of_imdcall

Syntax

objectname.of_imdcall( )

Argument	Description
objectname	An instance of the appeon_nvo_db_update object.

Return value

none.

Usage

Appeon Immediate Call Label cannot be used alone, it must be used in Appeon Queue Labels.

With the Appeon Immediate Call Label, the first database operation statement will be sent to the server and executed immediately.

Code Example

```
gnv_appeonDbLabel.of_startretrievequeue()
dw_1.retrieve()
gnv_appeonDbLabel.of_imdcall()
SELECT ......INTO :var_1,:var_2...
IF var_1 > 0 THEN
  para = "ok"
ELSE
  para = "false"
END IF
dw_2.retrieve(para)
gnv_appeonDbLabel.of_endretrievequeue()
```

2.3.5.6 Appeon Update Label

Description

The Appeon Update Label is used to reduce the number of interactions with the server caused by "interrelated updates".

Controls

appeon_nvo_db_update object

Associated functions

of_update

Syntax

objectname.of_update (integer transactionflag, powerobject *obj_1*, powerobject *obj_2*)

objectname.of_update (integer transactionflag, powerobject *obj_1*, powerobject *obj_2*, powerobject *obj_3*)

objectname.of_update (integer transactionflag, powerobject *obj_1*, powerobject *obj_2*, powerobject *obj_3*, powerobject *obj_4*)

objectname.of_update (powerobject *obj*)

objectname.of_update (powerobject *obj_1*, powerobject *obj_2*)

objectname.of_update (powerobject *obj_1*, powerobject *obj_2*, powerobject *obj_3*)

objectname.of_update (powerobject *obj_1*, powerobject *obj_2*, powerobject *obj_3*, powerobject *obj_4*)

Argument	Description
objectname	An instance of the appeon_nvo_db_update object.
transactionflag	0 - transaction is automatically committed; 1 - transaction is not automatically committed. Note: the of_update() function without this argument is preserved for compatibility.
obj	The name of the DataWindow, DataStore or DataWindowChild that needs to update.
obj_1	The name of the DataWindow, DataStore or DataWindowChild that needs to update.
obj_2	The name of the DataWindow, DataStore or DataWindowChild that needs to update.
obj_3	(optional) The name of the DataWindow, DataStore or DataWindowChild that needs to update.
obj_4	(optional) The name of the DataWindow, DataStore or DataWindowChild that needs to update.

Return value

Integer.	Return values are: 1 - Succeed in update -101 - Fail to update the first DataWindow/DataStore/DataWindowChild -102 - Fail to update the second DataWindow/DataStore/ DataWindowChild -103 - Fail to update the third DataWindow/DataStore/DataWindowChild -104 - Fail to update the fourth DataWindow/DataStore/ DataWindowChild

Usage

The update operations of the DataWindows, DataStores, or DataWindowChild will be submitted to the PowerServer together. If the operation of a DataWindow, DataStore or

DataWindowChild fails, PowerServer will stop processing the update operation. Users can also use transactionflag argument to control whether to commit or rollback the Database update.

The following script has the same function. However, by using the Appeon Update Label the number of client-server interactions is reduced to one.

Using Appeon Update Label

```
l_rtn = gnv_appeonDb.of_update(0, dw_1,dw_2)
IF l_rtn = 1 THEN
  Messagebox("Success","Update success!")
ELSEIF l_rtn= -102 THEN
  Messagebox("Failure","Update all failure!")
ELSE
  Messagebox("Failure","Update dw_1 failure!")
END IF
```

Without using Appeon Update Label

```
IF dw_1.Update() = 1 THEN
  IF dw_2.Update() = 1 THEN
    COMMIT;
    Messagebox("Success","Update success!")
  ELSE
    ROLLBACK;
    Messagebox("Failure","Update all failure!")
  END IF
ELSE
  ROLLBACK;
  Messagebox("Failure","Update dw_1 failure!")
END IF
```

2.3.6 Calling EJB Component

In order to call EJB components in applications deployed with Appeon, Appeon provides its own EJB solution which includes a customized object (EJBObject object in Appeon Workarounds PBL), a DLL (eonejbclient.dll), and a bridge (Appeon Bridge). With this EJB solution you can call an EJB component in both the PowerBuilder Client/Server application and the deployed Appeon application.

Compared to the EJB solution of PowerBuilder, Appeon EJB solution can support more complex parameters such as Structure; and there is no need to generate the proxy object in PowerBuilder.

eonejbclient.dll -- Located in the same folder as the Appeon Workarounds PBL, this DLL is used by the PowerBuilder Client/Server application only, not by the deployed Appeon application. If you want to make the PowerBuilder Client/Server application call the EJB component successfully using the same Appeon EJB solution as the deployed application uses, make sure this DLL is included in the PowerBuilder application directory on the client machine.

2.3.6.1 Supported application server

The Appeon EJB solution supports calling EJB 1.x/2.x/3.x components deployed to the J2EE-compliant application servers only (no .NET IIS application server). The certified supported application server includes WildFly 10, JBoss EAP 6.x, WebSphere 7/8/8.5, & WebLogic 12c.

2.3.6.2 Generate the EJB proxy

JBoss application server requires no EJB proxy generated, but WebLogic and WebSphere application server requires you to generate the EJB proxy and copy the generated EJB proxy to Appeon Bridge.

To generate the EJB proxy in the WebSphere application server:

1. Run the DOS command window as an administrator, and then change the directory to *IBM\WebSphere\AppServer\bin*.
2. Execute this command:

```
createEJBStubs.bat name of the JAR file or EAR file
```

To generate the EJB proxy in the WebLogic application server:

1. Run the DOS command window as an administrator, and then change the directory to *domains\base_domain\bin*.
2. Execute this command:

```
setDomainEnv.cmd
```

3. Change the directory back to *domains\base_domain\bin*, and execute this command:

```
java weblogic.appc name of the JAR file or EAR file
```

The proxy functions will be automatically generated into the JAR file and the JAR file that contains the proxy functions is what we call the EJB proxy. Follow the next section to copy/bundle the EJB proxy to the Appeon Bridge.

2.3.6.3 Bundle EJB proxy with Appeon Bridge

Introduction to Appeon Bridge

Appeon Bridge is a standard J2EE-compliant Web application that can be deployed to any J2EE compliant application server regardless of whether EJB components exist on the server. It functions as the medium between the clients and EJB components, and can be deployed by installing the appeonbridge.war file, which is located in *%appeon%\plugin\appeonbridege* under the PowerServer installation folder.

Bundle EJB proxy with Appeon Bridge

Before deploying Appeon Bridge to the application server, you must bundle the EJB proxy (JAR file) with Appeon Bridge to implement the communication to EJB components.

Three ways of bundling the EJB proxy with Appeon Bridge:

- Method 1: Add the proxy to the **lib** directory in the **appeonbridge.war** file. Since the **appeonbridge.war** file is a ZIP file, you can check the directory in the WAR file with a third-party tool such as WinZip.

  1. Navigate to *%appeon%\plugin\appeonbridge* folder and open **appeonbridge.war** with the third-party tool such as WinZip. %appeon% indicates the installation folder of the PowerServer.

2. Navigate to the folder where the EJB proxy is located.
3. Copy the entire folder of the EJB proxy to the *appeonbridge.war\WEB-INF\lib* directory (you will need to create the **lib** folder first).

This method does not work for EJB 3.x in JBoss; try the second method if you are using EJB 3.x in JBoss.

- Method 2: Add the classes of the EJB component to the **classes** directory in the **appeonbridge.war** file.

  1. Navigate to *%appeon%\plugin\appeonbridge* folder and open **appeonbridge.war** with the third-party tool such as WinZip. %appeon% indicates the installation folder of the PowerServer.
  2. Navigate to the folder where the EJB classes are located.
  3. Copy the entire folder of the EJB classes to the *appeonbridge.war\WEB-INF\classes* directory.

- Method 3: Copy the EJB proxy to the **lib** directory in the Java Web server, and add the **lib** directory (where the proxy is stored) to the CLASSPATH environment variable of the machine that hosts the Java Web server.

2.3.6.4 Deploy to application server

Now you are ready to deploy the following files to the application server one by one:

- Appeon Bridge (appeonbridge.war) which includes the EJB proxy. View Generate the EJB proxy for how to generate the EJB proxy and Bundle EJB proxy with Appeon Bridge for how to copy the EJB proxy into appeonbridge.war.
- Original JAR file for the EJB component without the EJB proxy functions.
- PowerBuilder application that calls the EJB component. View Call EJB component in PowerBuilder application for how to write PowerScript to call the EJB component functions.

Deploying any of the above files is the same as deploying the file server. Please refer to the corresponding **Deploying the Appeon File Server** section under Configuring and deploying the file server. (Or you can refer to documents provided by the corresponding server vendors for the deployment instructions, as deploying the above file is the same as deploying any other applications.)

2.3.6.5 Call EJB component in PowerBuilder application

This section guides you through writing PowerScript to call the EJB component. Appeon provides an object called EJBObject in the Appeon Workarounds PBL to call the EJB component, and provides Appeon Bridge to communicate with the EJB component.

Step 1: Add the Appeon Workarounds PBL to the Library Search Path of your PowerBuilder application.

Step 2: Declare the EJBObject object.

Step 3: Configure the connection string, call the EJBObject connectserver function to connect with Appeon Bridge on the application server.

The connection string varies according to the application server. View connectserver to determine the connection string.

Step 4: Create the instance for EJB component using EJBObject lookupjndi function or createremoteinstance function.

Note: For EJB 3.x, **lookupjndi** function works exactly as the **createremoteinstance** function; both functions can directly return the corresponding instance session handle; while for EJB 2.x, **lookupjndi** function will need to obtain the home interface of the EJB component first, for example, ls_msg = lo_ejb.lookupjndi ("TestSBeanLess",ref ll_homeHandle), before it can return the corresponding instance handle.

The JNDI name has to be accurate, and it varies according to the application server. For WebSphere, you will also need to set the JNDI name first in the WebSphere console. View JNDI Name for more information.

Step 5: If the function to be called contains parameter(s), call the EJBObject registering functions to register the parameter.

Step 6: Call the EJBObject invoking component functions to invoke the EJB component method.

Step 7: Destroy the instance and disconnect from the application server.

Below is a PowerScript code example that calls an EJB 3.x component:

```
ejbobject lo_ejb   //Declare EJBobject
long ll_bean = 0  //For receiving instance's handle
string ls_prop[5]
string ls_serurl,ls_msg,ls_msg1,ls_parm
//appeonbridge URL
ls_serurl = "http://"+ip+":"+port1+"/appeonbridge/Dispatch"
ls_prop[1]= is_appname //the deployed Appeon app name
ls_prop[2]="javax.naming.Context.INITIAL_CONTEXT_FACTORY=
 'org.jboss.as.naming.InitialContextFactory'"
ls_prop[3]="javax.naming.Context.PROVIDER_URL='remote://"+ip+":"+port2+"'"
ls_prop[4]= "username="
ls_prop[5]= "password="
ls_msg = lo_ejb.connectserver(ls_serurl, ls_prop) //Connect with server
if ls_msg = "" then
  //Connection is successful
else
  MessageBox("Not connected!",ls_msg)
end if
ls_jndi = sle_54.text //JNDI name has to be accurate otherwise instance will fail
 to create
//EJB 3.x supports passing any character in the third parameter
ls_msg = lo_ejb.createremoteinstance(ls_jndi,"AllDataType","sayHello", ref ll_bean)
if ls_msg = "" then
  //Instance is created successfully
  sle_53.text = string(ll_bean)
else
   MessageBox("createremoteinstance Failed!",ls_msg)
end if
ls_parm = "127.0.0.1"
//Register the parameter that will be called by the EJB functions
ls_msg = lo_ejb.regstring(ls_parm)
```

```
if ls_msg = "" then
else
   MessageBox("Regstring Failed !",ls_msg)
end if
//Call the function. Function name is case sensitive
ls_msg = lo_ejb.invokeretstring(ll_bean,"sayHello1",true, ref ls_msg1)
if ls_msg = "" then
  //Called successfully
  sle_52.text = ls_msg1
else
  MessageBox("invokestring Failed!",ls_msg)
end if
//Disconnect from server
ls_msg = lo_ejb.destroyremoteinstance(ll_bean)
if ls_msg <> '' then
 MessageBox("destroyremoteinstance Failed!",ls_msg)
end if
ls_msg = lo_ejb.disconnection( )
if ls_msg <> '' then
 MessageBox("disconnection Failed!",ls_msg)
end if
```

2.3.6.5.1 JNDI name

Different application servers have different JNDI names.

JNDI name for JBoss

To call EJB 3.x in JBoss, the JNDI name should be in this format: ejb:" + appName + "/" + moduleName + "/" + distinctName + "/" + beanName + "!" + viewClassName

appName: name of the EAR file. If it is JAR file instead of EAR file, then leave this empty.

moduleName: name of the deployed JAR file; file extension should not be contained here, for example, HelloWorld.

distinctName: if no such name is defined, then leave this empty.

beanName: name of the implementation class.

viewClassName: name of the interface and package, for example, com.ibytecode.business.HelloWorld.

JNDI name for WebSphere

To call EJB in WebSphere, you will need to define the JNDI name by yourself:

1. Log into the WebSphere Administration console.
2. In the left pane, expand **Applications** > **Application Types**, and then select **WebSphere enterprise applications**.
3. In the right pane, click the name of the EJB component, and then click **Bind EJB Business** or **EJB JNDI Names**.
4. Define the name in the **JNDI Name** text box.

JNDI name for WebLogic

To call EJB in WebLogic 12c, you can find out the JNDI name here:

1. Log into the WebLogic Server Administration Console.
2. In the left pane, expand **Environment** > **Servers**.
3. On the **Server summary** page, click the name of the server.
4. On the server **Settings** page, click **View JNDI Tree**.

   The JNDI tree is displayed in a new browser window.
5. In the left pane for the JNDI tree, expand the **java:global** node, find the EJB component, expand the component to find the class, and select the class.

   The **Bind Name** displayed in the right page is the JNDI name you are looking for.

2.3.6.5.2 EJBObject object

Appeon EJBObject object implements the interaction between client and Appeon Bridge. To use this object, you need to load **appeon_workaround.pbl** into your application.

EJBObject object provides the following functions to perform the relevant actions:

Function	Description
ConnectServer function DisConnection function	Connecting and disconnecting to Appeon Bridge
LookUpJndi function	Obtaining the home interface of an EJB component
CreateRemoteInstance function DestroyRemoteInstance function	Creating and destroying the instance for an EJB component
Registering parameter functions	Registering parameters
Invoking component functions	Invoking EJB components
InitLocalLanguage function	Setting the language of the error message

ConnectServer function

Description

Connects a client application to Appeon Bridge.

Syntax

EJBObject.ConnectServer(string *url*, string *properties[]*)

Parameter	Description
EJBobject	A reference of an EJBobject.

Parameter	Description
url	URL with port, where Appeon Bridge is installed. Different application servers have different URL. See below for examples.
properties[]	Properties of Appeon Bridge.

Return value

String.

Returns empty string ("") if it succeeds.

Here are a few examples of the URL for different application servers:

Example for EJB 1.x/2.x in JBoss:

```
String ls_serurl,ls_prop[5]
ls_serurl  = "http://127.0.0.1:8080/appeonbridge/Dispatch"
ls_prop[1]= "testappeon" //name of the deployed Appeon application
ls_prop[2]=
 "javax.naming.Context.INITIAL_CONTEXT_FACTORY='org.jnp.interfaces.NamingContextFactory'"
ls_prop[3]= "javax.naming.Context.PROVIDER_URL='jnp://127.0.0.1:8080'"
ls_prop[4]= "username="
ls_prop[5]= "password="
```

Example for EJB 3.x in JBoss:

```
String ls_serurl,ls_prop[5]
ls_serurl= "http://127.0.0.1:8080/appeonbridge/Dispatch"
ls_prop[1]= "testappeon" //name of the deployed Appeon application
ls_prop[2]= "javax.naming.Context.INITIAL_CONTEXT_FACTORY=
 'org.jboss.as.naming.InitialContextFactory'"
ls_prop[3]= "javax.naming.Context.PROVIDER_URL= 'remote://127.0.0.1:4447'"//default
 port is 4447.
//Port number can be configured in {$Home_DIR}\standalone\configuration
\standalone.xml.
ls_prop[4]= "username="
ls_prop[5]= "password="
```

Example for WebSphere:

```
String ls_serurl,ls_prop[5]
ls_serurl  = "http://127.0.0.1:9080/appeonbridge/Dispatch"
ls_prop[1]= "testappeon" //name of the deployed Appeon application
ls_prop[2]="javax.naming.Context.INITIAL_CONTEXT_FACTORY='com.ibm.websphere.naming.WsnInitialCont
ls_prop[3]= "javax.naming.Context.PROVIDER_URL='iiop://127.0.0.1:2809'"//default
 port is 2809.
//Port number can be obtained or changed in WebSphere console > server1 >
 BOOTSTRAP_ADDRESS.
ls_prop[4]= "username=admin"
ls_prop[5]= "password=admin"
```

Example for WebLogic:

```
String ls_serurl,ls_prop[5]
ls_serurl  = "http://127.0.0.1:7001/appeonbridge/Dispatch"
ls_prop[1]= "testappeon" //name of the deployed Appeon application
ls_prop[2]=
 "javax.naming.Context.INITIAL_CONTEXT_FACTORY='weblogic.jndi.WLInitialContextFactory'"
ls_prop[3]= "javax.naming.Context.PROVIDER_URL='t3://127.0.0.1:7001'"
ls_prop[4]= "username=weblogic"
ls_prop[5]= "password=appeon123"
```

DisConnection function

Description

Disconnects a client application from Appeon Bridge.

Syntax

EJBObject.DisConnection()

Parameter	Description
EJBobject	A reference of an EJBobject.

Return value

String.

Returns empty string ("") if it succeeds.

LookUpJndi function

Description

Obtains the home interface of an EJB component in order to create an instance for the component.

Syntax

EJBObject.lookupjndi (string *jndiname*, ref long *objid*)

Parameter	Description
EJBObject	A reference of an EJBobject.
jndiname	The JNDI name of the EJB component.
objid	The handle to the EJB home interface.

Return value

String.

Returns empty string ("") if it succeeds.

InitLocalLanguage function

Description

Sets the language of the error message in PowerBuilder IDE.

Syntax

EJBObject.initlocallanguage (long *nlocalcode*)

Parameter	Description
EJBObject	A reference of an EJBobject.
nlocalcode	A long value representing different languages.

Parameter	Description
	0 - English (default) 1 - Japanese 2 - Korean 3 - Simplified Chinese 4 - Traditional Chinese

Return value

Long.

Invoking component functions

Invokeretblob	Invokeretblobarray	Invokeretbool	Invokeretboolarray
Invokeretchar	Invokeretchararray	Invokeretdate	Invokeretdatearray
Invokeretdatetime	Invokeretdatetimearray	Invokeretdouble	Invokeretdoublearray
Invokeretint	Invokeretintarray	Invokeretlong	Invokeretlongarray
Invokeretreal	Invokeretrealarray	Invokeretstring	Invokeretstringarray
Invokerettime	Invokerettimearray	Invokeretuint	Invokeretuintarray
Invokeretulong	Invokeretulongarray	Invokeretstru	Invokeretstruarray
Invokeretvoid			

Description

Invoke an EJB component which returns a particular data type. All methods will share the same parameters, syntax and return value.

Syntax

string EJBObject.Invokeblob(long *objid*, string *methodname*, boolean *autoremove*, ref blob *retval*)

Parameter	Description
EJBObject	A reference of an EJBobject.
objid	The handle to the component method.
methodname	The name of the invoking component method.
autoremove	Unsupported. Input false.
retval	The return value of invoking the EJB component method. This parameter does not provided in the Invokeretvoid method. The data type of the retval argument keeps the consistence with the data type used in the invoking methods, except the Invokeretstru

Parameter	Description
	and Invokeretstruarray, the data type of the retval arguments for the two methods are both blob.

Return value

String.

Returns empty string ("") if it succeeds.

Usage

Variables cannot be null for structure and array.

For a structure to be registered, variables can be:

1. char, string, boolean, int, unit, long, ulong, real, double, datetime, date or time.
2. an array of the above types. The maximum dimension is 3.
3. a structure or a 1-dimensional structure array. And the array must be a fixed array.

For a return value of structure type, variables can be:

1. char, string, boolean, blob, int, unit, long, ulong, real, double, datetime, date or time.
2. a multidimensional array of the above types.
3. a structure or a 1-dimensional structure array. And the array must be a fixed array.

Registering parameter functions

Register functions are provided to register parameters with different data type. Except RegStruct and RegstructArray, all functions will share the same parameters, syntax, return value.

RegChar	RegCharArray	RegDate	RegDateArray
RegDateTime	RegDateTimeArray	RegDouble	RegDoubleArray
RegInt	RegIntArray	RegLong	RegLongArray
RegReal	RegRealArray	RegString	RegStringArray
RegBlob	RegBlobArray	RegBool	RegBoolArray
RegTime	RegTimeArray	RegUInt	RegUIntArray
RegULong	RegULongArray	RegStruct	RegStructArray

Description

Registers a parameter with a certain data type. Syntax below takes RegBlob as an example.

Syntax

EJBObject.RegBlob(blob *data*)

Parameter	Description
EJBObject	A reference of an EJBobject.
data	The parameter to be registered. Its name can be user-defined, but its type must be consistent with the data type specified in the Register method.

Return value

String.

Returns empty string ("") if it succeeds.

RegStruct and RegStructArray functions

Description

Registers a structure or structure array. The two functions will share the same parameters, syntax, return value. Syntax below takes regstructarrary as an example.

Syntax

EJBObject.regstructarrary (any *data []*, string *javaclassname*, readonly classdefinition *cdef*)

Parameter	Description
EJBObject	A reference of an EJBobject.
data	The parameter to be registered and the name is user-defined.
javaclassname	The name of the corresponding Java class in application server.
cdef	ClassDefinition property of structure.

Return value

String.

Returns empty string ("") if it succeeds.

Usage

Variables cannot be null for structure and array.

For a structure to be registered, variables can be:

1. char, string, bool, int, unit, long, ulong, real, double, datetime, date or time.
2. an array of the above types. The maximum dimension is 3.
3. a structure or a 1-dimensional structure array. And the array must be a fixed array.
4. (Only for RegStruct method) For the javaclassname parameter, input the full name of the Java class on EJB server corresponding to the structure you defined in PowerBuilder. For example, a.b.c.d.myclassName.

CreateRemoteInstance function

Description

Creates the instance for an EJB component.

Syntax

EJBObject.CreateRemoteInstance(string *jndiname*, string *homename*, string *methodname*, ref long *beanid*)

Parameter	Description
EJBObject	A reference of an EJBobject.
jndiname	The JNDI name of the EJB component.
homename	The name of the home interface of an EJB component.
methodname	The name of the method.
beanid	The handle to the EJB component.

Return value

String.

Returns empty string ("") if it succeeds.

DestroyRemoteInstance function

Description

Destroys the instance for an EJB component.

Syntax

EJBObject.DestroyRemoteInstance(long *objid*)

Parameter	Description
EJBObject	A reference of an EJBobject.
objid	The handle to the EJB component.

Return value

String.

Returns empty string ("") if it succeeds.

2.3.6.5.3 Appeon requirements for EJB development

1. Appeon Bridge maps datatypes (except structure) between Java and PowerBuilder is shown as below.

PowerBuilder data types	Java data types
Char	char

PowerBuilder data types	Java data types
String	String
Boolean	boolean
Int	short
Unit	int
Long	int
Ulong	long
LongLong	long
Real	float
Double	double
Decimal	java.math.BigDecimal
Number	double
Timestamp	java.sql.Timestamp
Datetime	java.sql.Timestamp
Date	java.util.Date
Time	java.sql.Time
Blob	byte[]

2. With Appeon EJB solution, Structure data can be passed when invoking EJB components. To implement this, you need to define a Java class in the EJB components. There are two necessary elements in Java Class: 1) **private static String PBMap[]** and 2) **implementing java.io.Serializable interface**. In PBMap array you need to map the members with the identical order and datatype to a PowerBuilder Structure.

   Following is an example of defining a Java class (please note that the member variables should be in lower case.)

```
package test;
import java.io.Serializable;

public class Simple implements Serializable {
  private short l_int;
  private boolean b_bool;
  private String s_string;

  private static String PBMap[] = {"l_int", "b_bool", "s_string"};

  public String[] getPBMap() {
    return PBMap;
  }

  public boolean isB_bool() {
    return b_bool;
  }

  public short getL_int() {
    return l_int;
  }

  public String getS_string() {
```

```
    return s_string;
  }

  public void setB_bool(boolean b_bool) {
    this.b_bool = b_bool;
  }

  public void setL_int(short l_int) {
    this.l_int = l_int;
  }

  public void setS_string(String s_string) {
    this.s_string = s_string;
  }
}
```

2.3.7 Calling .NET/COM server components (.NET only)

Applies to

PowerServer for .NET.

Supported server component types

- .NET components: All valid .NET components, including executable files (.exe) and DLL files (.dll).

  Supported parameters: primitive type parameters, such as int, vlong char, and boolean. Non-primitive type parameters, such as class, are unsupported.

  Supports reference parameters.

- COM components: COM/COM+ components

  Supported parameters: primitive type parameters, such as byte, int, long, and float.

  Supports reference parameters.

Description

To call .NET/COM components, Appeon provides a non-autoinstantiated NVO - AppeonDotNetComponent - as the proxy object to call the server-side components. The user can either create a local instance of AppeonDotNetComponent for each server component, or directly use an existing instance of AppeonDotNetComponent. The user must specify the properties of the instance, such as the component type, the library name and the class name, to bind the instance with the server component, or change the instance properties during runtime to dynamically bind with a different component.

It provides a universal single interface and a set of parameters which determines which component and methods will be called.

Note:

The script to call AppeonDotNetComponent takes effect only after the PowerBuilder application is deployed, and has no effect when the PowerBuilder application is run.

Register

The COM component must be registered using the regsvr32 tool.

Storage location

The components must reside in the %appeon%/AEM/components folder on the PowerServer machine. You only need to place the .tlb library files and .dll files of the COM components to the folder. %appeon% indicates the installation directory of PowerServer.

2.3.7.1 AppeonDotNetComponent object

2.3.7.1.1 Properties

Properties for AppeonDotNetComponent.

Properties	Type	Description
ComponentType	String	The type of the component to be called. "1" indicates a .NET Assembly to be called. "2" indicates an unmanaged-code COM component to be called. "3" indicates a managed-code COM component to be called. "4" indicates a built-in Appeon Workaround .NET Assembly to be called.
TypeLib	String	The name of the component library. PowerServer uses this name to find the component.
ClassDescript	String	The class name.
ReturnValue	Any	Read-only. The return value of functions. The value and value type varies from function to function.
ErrorText	String	Read-only. The error message of functions. The message varies from function to function. Empty string if no error.

2.3.7.1.2 Functions

of_execinterface

Description

Calls the function in the binding component.

Syntax

of_execinterface (string *`interfacename`* {, ref any *`paralist`*[]})

Argument	Description
`interfacename`	The name of the component function.
`paralist[]`	Optional. Arrays of Any type. Specifies the parameter arrays for the component function.

Return value

Long.

0 – Call succeeded. Gets the value from the ReturnValue property of the proxy object.

-1 – Call failed. Gets the error message from the ErrorText property of the proxy object.

Usage

Before calling this function, use the proxy object properties to bind with the target component. If the component function contains no parameters, simply specify the function name. If the component function contains parameters, define an Any type array before the call, then place the argument to the array, finally pass the array as the second parameter of the function.

Examples

Example 1: the interface contains no parameters.

```
AppeonDotNetComponent lu_apf

lu_apf = create AppeonDotNetComponent lu_apf
lu_apf.ComponentType = "2"
lu_apf.TypeLib = "test.dll"
lu_apf.ClassDescript = "testclass"

ll_ret = lu_apf.of_ExecInterface("test")
```

Example 2: the interface contains four parameters, their types are: string, int, long, and string.

```
// Define the array variable
AppeonDotNetComponent lu_apf
any la_1[]

la_1[1] = "Appeon"
la_1[2] = 100
la_1[3] = 256
la_1[4] = "Sybase"

lu_apf = create AppeonDotNetComponent lu_apf
lu_apf.ComponentType = "1"
lu_apf.TypeLib = "testdotnet.dll"
lu_apf.ClassDescript = "interface1"

ll_ret = lu_apf.of_ExecInterface("test_dotnet", la_1)
```

2.3.7.1.3 Events

Constructor

Description

It will be triggered when you create an instance from a user-defined proxy object inherited from AppeonDotNetComponent.

Event ID

pbm_constructor

Argument

None

Return values

Long

Usage

Do not write scripts to this event directly, because the scripts will be abandoned when the application is deployed. Instead, define and inherit an object from AppeonDotNetComponent, and add the scripts to the Constructor event of the new object. The usage is the same as that of the PowerBuilder system object. For example, you can initialize the property value of this event, or define relevant information objects.

Destructor

Description

It will be triggered when you explicitly call Destroy to destroy the instance of a user-defined proxy object inherited from AppeonDotNetComponent.

Event ID

pbm_destructor

Argument

None

Return value

Long

Usage

Do not write scripts to this event directly, because the scripts will be abandoned when the application is deployed. Instead, define and inherit an object from AppeonDotNetComponent, and add the scripts to the Destructor event of the new object. The usage is the same as that of the PowerBuilder system object. For example, you can add scripts to release the instances related with the proxy object.

2.3.7.1.4 Code Examples

Example 1:

```
long lRet
int iResult
string strError
appeondotnetcomponent comcaller

// create appeondotnetcomponent instance and set properties

comcaller = create appeondotnetcomponent
comcaller.componenttype = '1'
comcaller.typelib = 'DotNetDll.dll'
comcaller.classdescript = 'DotNetClass'

// invoke component method

lRet = comcaller.of_execinterface("GetInt")
if lRet = 0 then
iResult = comcaller.ReturnValue
else
strError = comcaller.ErrorText
end if

// Bind with a component and call the component method

comcaller.componenttype = '2'
```

```
comcaller.typelib = 'comfordotnet.dll'
comcaller.classdescript = 'ifdotnet'
comcaller.of_execinterface("getint")
```

Example 2:

```
// Call a method with reference parameters

any paralist[]
long refparam1 = 32764
long refparam2 = 32763

paralist[1] = refparam1
paralist[2] = refparam2

comcaller.componenttype = '1'
comcaller.typelib = 'DotNetDll.dll'
comcaller.classdescript = 'DotNetClass'
comcaller.of_execinterface("GetIntAndRefInt",paralist)

refparam1 = paralist[1]
rafparam2 = paralist[2]
```

2.3.8 Calling Web Service

Description

To call Web services, Appeon provides a non-autoinstantiated NVO – AppeonWebServiceComponent – as the proxy object to call Web services. The user can either create a local instance of AppeonWebServiceComponent for Web service, or directly use an existing instance of AppeonWebServiceComponent. The user must specify the properties of the instance, such as the proxy type, the Web service location and the class name, to bind the instance with the Web service, or change the instance properties during runtime to dynamically bind with a different Web service.

It provides a universal single interface and a set of parameters which determines which Web service and methods will be called.

Note:

1. The script to call AppeonWebServiceComponent takes effect on both the Web and the mobile, and has no effect in the PowerBuilder application.

2. You need to restart the IIS after deploying a Web service with a new method.

2.3.8.1 appeonwebservicecomponent object

2.3.8.1.1 Properties

Properties for appeonwebservicecomponent.

Properties	Type	Description
CallType	String	The proxy type of the Web service to be called. "1" indicates the proxy type is Dynamic Proxy. "2" indicates the proxy type is DLL Proxy.

Properties	Type	Description
		PowerServer for J2EE editions support only CallType="1".
ProxyDllOrUrl	String	If CallType="1", it indicates the URL of the Web service to be called, for example, http://localhost/webservice.asmx (the string "http://" is required); If CallType="2", it indicates the DLL name of the proxy used by the Web service to be called.
ClassDescriptor	String	If CallType="1", it can be null; If CallType="2", it indicates the class name of the nonvisual object deployed as the Web service on the server side.
ReturnValue	Any	Read-only. The return value of functions. The value and value type varies from function to function.
ErrorText	String	Read-only. The error message of functions. The message varies from function to function. Empty string if no error.

2.3.8.1.2 Functions

of_callwebservice

Description

Calls the function in the binding Web service.

Syntax

of_callwebservice (value string *methodname* {, ref any *paralist[]*})

Argument	Description
methodname	The name of the Web service method.
paralist[]	Optional. Arrays of Any type. Specifies the parameter arrays for the Web service function.

Return value

Long.

0 – Call succeeded. Gets the value from the ReturnValue property of the proxy object.

-1 – Call failed. Gets the error message from the ErrorText property of the proxy object.

Usage

Before calling this function, use the proxy object properties to bind with the target Web service. If the Web service method contains no parameters, simply specify the method name. If the Web service method contains parameters, define an Any type array before the call, then place the argument to the array, finally pass the array as the second parameter of the function.

Examples

Example 1: the interface contains no parameters.

```
appeonwebservicecomponent caller
caller= create appeonwebservicecomponent
```

```
caller.calltype="1"
caller.proxydllorurl="http://localhost/webservice.asmx"
caller.classdescript=""
integer IRet
IRet=caller.of_callwebservice("GetUserName")
```

Example 2: the interface contains two parameters, their types are any.

```
any paralist[]
appeonwebservicecomponent caller
caller=create appeonwebservicecomponent
caller.calltype="1"
caller.proxydllorurl="http://localhost/webservice.asmx"
caller.classdescript=""
paralist[1]="param1"
paralist[2]="param2"
IRet=caller.of_callwebservice("GetUserName",paralist)
```

2.3.8.1.3 Events

Constructor

Description

It will be triggered when you create an instance from a user-defined proxy object inherited from AppeonWebServiceComponent.

Event ID

pbm_constructor

Argument

None

Return values

Long

Usage

Do not write scripts to this event directly, because the scripts will be abandoned when the application is deployed. Instead, define and inherit an object from AppeonWebServiceComponent, and add the scripts to the Constructor event of the new object. The usage is the same as that of the PowerBuilder system object. For example, you can initialize the property value of this event, or define relevant information objects.

Destructor

Description

It will be triggered when you explicitly call Destroy to destroy the instance of a user-defined proxy object inherited from AppeonWebServiceComponent.

Event ID

pbm_destructor

Argument

None

Return value

Long

Usage

Do not write scripts to this event directly, because the scripts will be abandoned when the application is deployed. Instead, define and inherit an object from AppeonWebServiceComponent, and add the scripts to the Destructor event of the new object. The usage is the same as that of the PowerBuilder system object. For example, you can add scripts to release the instances related with the proxy object.

2.3.8.1.4 Code Examples

Example 1:

```
long IRet
int iResult
string strError
appeonwebservicecomponent caller

// create appeonwebservicecomponent instance and set properties

caller=create appeonwebservicecomponent

// if proxy type is DynamicProxy, the value of calltype is 1;
// if proxy type is DllProxy type, the value of calltype is 2.
// PowerServer for J2EE editions support only CallType="1".
// PowerServer for .NET edition supports both.

caller.calltype="1"

// since proxy type is DynamicProxy,the value of proxydllorurl is the address of
// the webservice to be called;or else, the value of proxydllorurl is the name
// of DLL.
caller.proxydllorurl="http://localhost/webservice.asmx"
caller.classdescript=""

// Invoke webservice method

IRet=caller.of_callwebservice("GetUserName")
if IRet=0 then
 iResult=caller.ReturnValue
else
 strError=caller.ErrorText
end if
```

Example 2:

```
long IRet
int iResult
string strError
any paralist[]
appeonwebservicecomponent caller

// create appeonwebservice instance and set properties

caller=create appeonwebservicecomponent
caller.calltype="1"
caller.proxydllorurl="http://localhost/webservice.asmx"
caller.classdescript=""
paralist[1]="param1"
paralist[2]="param2"

// invoke webservice method
```

```
IRet=caller.of_callwebservice("GetUserName",paralist)
```

Example 3:

```
long IRet
int iResult
string strError
appeonwebservicecomponent caller

// create appeonwebservicecomponent instance and set properties

caller=create appeonwebservicecomponent
caller.calltype="2"
caller.proxydllorurl="DotDllForJava"
caller.classdescript="MyJavaWebService"

// invoke webservice method

IRet=caller.of_callwebservice("GetUserName")
if IRet=0 then
 iResult=caller.ReturnValue
else
 strError=caller.ErrorText
end if
```

2.3.9 Appeon Text Controls

2.3.9.1 eon_singlelineedit, eon_statictext, & eon_editmask

2.3.9.1.1 Overview

Appeon provides additional properties and events to help developers take advantage of SingleLineEdit control, StaticText control, and EditMask control:

- In order to align text centered vertically, Appeon provides a new property (integer `ii_vertical`) and a user event (`ue_setvertical`) for the **SingleLineEdit** control, **StaticText** control, and **EditMask** control.
- To allow developers to take control of the type of the virtual keyboard on iOS and Android, Appeon provides a new property (integer `ii_keyboardtype`) for the **EditMask** and **SingleLineEdit** control.

To use the new properties or user event, you will need to call the **eon_singlelineeditex**, **eon_statictextex**, or **eon_editmaskex** objects in the Appeon Workarounds PBL.

Except for the new properties and user event, the objects have the exact same functions, events, and properties as the PowerBuilder SingleLineEdit control, StaticText control, and EditMask control.

2.3.9.1.2 Properties

integer ii_vertical

Sets the text alignment. When `ii_vertical` is set to 1, text in **eon_singlelineeditex**, **eon_statictextex**, or **eon_editmaskex** is centered vertically; when set to 0, text alignment is consistent with that in PowerBuilder.

integer ii_keyboardtype

This property is applicable to **eon_editmaskex** and **eon_singlelineeditex** on mobile platform only.

Sets the type of the keyboard that displays when **eon_editmaskex** or **eon_singlelineeditex** gets focus: 0 - alphabetical keyboard, 1 - numerical keyboard, 2 - URL keyboard.

2 is unsupported on Android devices, thus 0 will be used instead if 2 is set for Android devices.

2.3.9.1.3 Events

ue_setvertical

Description

Sets the `ii_vertical` property for the object.

Syntax

`singlelineedit``.ue_setvertical` (integer *`ai_vertical`*)

`statictext``.ue_setvertical` (integer *`ai_vertical`*)

`editmask``.ue_setvertical` (integer *`ai_vertical`*)

Argument Type	Argument	Description
eon_mobile_singlelineeditex	*`singlelineedit`*	The name of the eon_singlelineeditex object.
eon_mobile_statictextex	*`statictext`*	The name of the eon_statictextex object.
eon_mobile_editmaskex	*`editmask`*	The name of the eon_editmaskex object.
integer	*`ai_vertical`*	The value of the vertical property (`ii_vertical`) for the object.

Return value

Integer

1 - the object is centered vertically.

0 - text alignment of the object is consistent with PowerBuilder.

2.3.10 Creating Data Source Dynamically

2.3.10.1 Overview

Data source can be predefined/modified/deleted in AEM (refer to Section 5.3.3.2, “Data Source” in *PowerServer Configuration Guide for .NET* for details), or dynamically created/modified/deleted via **appeondconfigdatasource** object in Appeon Workarounds PBLs. But unlike creating data source in AEM, you could only dynamically create data source for

the following database types: Microsoft SQL Server, Teradata, MySQL, SAP ASE, and PostgreSQL (data sources for Oracle, IBM Informix, SAP HANA, and SAP SQL Anywhere cannot be created via **appeondconfigdatasource** object; they can only be created in AEM).

Note that you can only call **appeondconfigdatasource** object to dynamically create data source on the PowerServer for .NET version (not J2EE version).

2.3.10.2 appeondconfigdatasource

2.3.10.2.1 of_adddatasource function

Description

Create a data source with the specified parameters.

Syntax

`of_adddatasource` (eon_str_connection *`astr_connection`*)

Argument	Description
`astr_connection`	The parameters for the data source. Refer to eon_str_connection structure for the parameter list.

Return value

Integer

0 - Success.

-1 - The parameters are incorrect. Failed to create the data source.

1 - The data source name already exists.

2 - The database type is not supported.

3 - The database connection failed.

2.3.10.2.2 of_deletedatasource function

Description

Delete the specified data source.

Syntax

`of_deletedatasource` (string *`as_datasourcename`*) returns integer

Argument	Description
`as_datasourcename`	The name of the data source to be deleted.

Return value

Integer

0 - Success.

-1 - The parameters are incorrect. Failed to delete the data source.

1 - The data source name does not exist. Failed to delete the data source.

2.3.10.2.3 of_editdatasource function

Description

Modify the parameters of the specified data source.

Syntax

`of_editdatasource` (eon_str_connection *`astr_connection`*)

Argument	Description
`astr_connection`	The parameters of the specified data source. Refer to eon_str_connection structure for the parameter list.

Return value

Integer

0 - Success.

-1 - The parameters are incorrect. Failed to modify the data source.

1 - The data source name does not exist.

2 - The database type is not supported.

3 - The database connection failed.

2.3.10.2.4 eon_str_connection structure

Description

Struct.

The parameters of the data source to be created/modified.

Type	Variable Name	Description
string	*`s_dbtype`*	The database type of the data source. Supported values are: sqlserver, teradata, mysql, ase, postgresql, db2
string	*`s_datasourcename`*	The name of the data source to be created/modified.
string	*`s_server`*	The machine name or IP address of the database server.
string	*`s_port`*	The port number of the database server.
string	*`s_database`*	The database name.
string	*`s_username`*	The user name of the database.
string	*`s_password`*	The password of the database.
string	*`s_maxpool`*	The maximum number of connections PowerServer opens and pools on startup.
string	*`s_minpool`*	The minimum number of connections PowerServer opens and pools on startup.

Type	Variable Name	Description
string	*s_pooling*	Whether to use the connection pool. Supported values are: true, false.
string	*s_connectiontimeout*	The timeout period (in seconds) for the connection.
string	*s_commandtimeout*	The timeout period (in seconds) for the commands.
string	*s_usedynamicconn*	Whether to enable dynamic database connection. Supported values are: true, false.

2.3.11 Accessing JSON-format Data

2.3.11.1 Overview

JSON is a light-weight data format for transmitting data in JavaScript and many other programing languages. Appeon provides two objects (eon_cjsonnode & eon_cjsonnodearray) to help developers easily handle the JSON-format data. These two objects are supported in PowerServer Web and PowerServer Mobile.

- eon_cjsonnode is for manipulating the JSON node (a collection of name/value pairs), including parsing names/values from a string, adding/removing name/value pairs, converting the node to a string etc. A JSON node can contain name/value pairs, JSON nodes, and JSON arrays.

  A JSON node begins with left brace ("{") and ends with right brace ("}"), and each name is followed by colon (":") and the name/value pairs are separated by comma (","), for example:

```
{
"firstname": "Brett",
"lastname": "Zhang",
"email": "brett.zhang@appeon.com"
}
```

- eon_cjsonnodearray is for manipulating the array values (an ordered list of values) of the JSON node, including parsing values from a string, adding/removing values, converting the array to a string etc. A JSON array can contain values, JSON nodes, and JSON arrays.

  A JSON array begins with left bracket ("[") and ends with right bracket ("]"), and values are separated by comma (","), for example:

```
{ "people": [
{"firstname": "Brett", "lastname": "Zhang", "email": "brett.zhang@appeon.com"},
{"firstname": "Jason", "lastname": "Li", "email": "jason.li@appeon.com"},
{"firstname": "Mary", "lastname": "Wang", "email": "mary.wang@appeon.com"}
] }
```

2.3.11.2 eon_cjsonnode

2.3.11.2.1 of_addkey

Description

Adds a key (name/value pair) under the current JSON node.

Syntax

`of_addkey` (string *`as_keyname`*, any *`aa_keyvalue`*)

Argument	Description
`as_keyname`	Name of the new key. When the name already exists, it will be overwritten.
`aa_keyvalue`	Value of the new key. Supported data type of the value: long, boolean, string, double, eon_cjsonnode, & eon_cjsonnodearray

Return value

None

2.3.11.2.2 of_clearnode

Description

Clears all of the keys (name/value pairs) under the current JSON node.

Syntax

`of_clearnode` ( )

Return value

None

2.3.11.2.3 of_deletekey

Description

Removes a key (name/value pair) from the current JSON node.

Syntax

`of_deletekey` (string *`as_keyname`*)

Argument	Description
`as_keyname`	Name of the key to be deleted.

Return value

None

2.3.11.2.4 of_getkeybyindex

Description

Gets the name of the key specified by the index.

Syntax

`of_getkeybyindex` (integer *`ai_index`*)

Argument	Description
ai_index	Value of the index pointing to a key.

Return value

String. The name of the key specified by the index.

Null if the key does not exist.

2.3.11.2.5 of_gettypebyindex

Description

Gets the data type of the value specified by the index.

Syntax

`of_gettypebyindex` (integer *`ai_index`*)

Argument	Description
ai_index	Value of the index pointing to a key.

Return value

Integer. Data type of the value specified by the index:

0 - null

1 - string

2 - number (including long and double)

3 - boolean

4 - eon_cjsonnodearray

5 - eon_cjsonnode

-1 - invalid value

2.3.11.2.6 of_gettypebykey

Description

Gets the data type of the value for the specified key name.

Syntax

`of_gettypebykey` (string *`as_keyname`*)

Argument	Description
as_keyname	Name of the key.

Return value

Integer. Data type of the value.

0 - null

1 - string

2 - number (including long and double)

3 - boolean

4 - eon_cjsonnodearray

5 - eon_cjsonnode

-1 - invalid value

2.3.11.2.7 of_load

Description

Loads the JSON-format string into the current JSON node.

Syntax

`of_load` (string *`as_jsonstring`*)

Argument	Description
`as_jsonstring`	The JSON-format string. Make sure the string complies with the JSON and JavaScript language standards: • The key name is case sensitive. • The key name must be included in double quotation marks. • The string value must be included in double quotation marks. • The boolean values (true/false) must be in lower case. • The boolean value, double value, and long value need not be included in double quotation marks.

Return value

Boolean

True - Success.

False - Failed to load.

2.3.11.2.8 of_setvalue

Description

Sets the value for the specified key.

Syntax

`of_setvalue` (string *`as_keyname`*, any *`aa_keyvalue`*)

Argument	Description
`as_keyname`	Name of the key.
`aa_keyvalue`	Value of the key.

Argument	Description
	Supported data types include: long, boolean, string, double, eon_cjsonnode, & eon_cjsonnodearray

Return value

None

2.3.11.2.9 of_size

Description

Gets the total number of the keys (name/value pairs) under the current JSON node.

Syntax

```
of_size()
```

Return value

Integer. The total amount of the keys.

-1 - It is called in PowerBuilder, or there is an error.

2.3.11.2.10 of_torawstring

Description

Converts the keys (name/value pairs) under the current node to a special JSON-format string. Different from the standard format, the key value in the special format will not be included in double quotation marks automatically. The developer use this function according to the need of excluding double quotes in a key value, for example, to generate the proper date format in a JSON string.

Syntax

```
of_torawstring()
```

Return value

String. The JSON-format string.

2.3.11.2.11 of_tostring

Description

Converts all of the keys (name/value pairs) under the current node to a standard JSON-format string.

Syntax

```
of_tostring()
```

Return value

String. The JSON-format string.

2.3.11.2.12 of_valuearray

Description

Gets the value (array) of the specified key.

Syntax

`of_valuearray` (string *`as_keyname`*)

Argument	Description
`as_keyname`	Name of the key.

Return value

eon_cjsonnodearray. The array value (eon_cjsonnodearray object) of the specified key.

Null if the specified key does not exist or the value is not an array.

2.3.11.2.13 of_valueboolean

Description

Gets the boolean value of the specified key.

Syntax

`of_valueboolean` (string *`as_keyname`*)

Argument	Description
`as_keyname`	Name of the key.

Return value

Boolean. The value of the specified key.

Null if the specified key does not exist or the value is not a boolean.

2.3.11.2.14 of_valuedouble

Description

Gets the double value of the specified key.

Syntax

`of_valuedouble` (string *`as_keyname`*)

Argument	Description
`as_keyname`	Name of the key.

Return value

Double. The value of the specified key.

Null if the specified key does not exist or the value is not a double.

2.3.11.2.15 of_valuelong

Description

Gets the long value of the specified key.

Syntax

`of_valuelong` (string *`as_keyname`*)

Argument	Description
`as_keyname`	Name of the key.

Return value

Long. The value of the specified key.

Null if the specified key does not exist or the value is not a long.

2.3.11.2.16 of_valuenode

Description

Gets the value (eon_cjsonnode object) of the specified key.

Syntax

`of_valuenode` (string *`as_keyname`*)

Argument	Description
`as_keyname`	Name of the key.

Return value

eon_cjsonnode. The value (eon_cjsonnode object) of the specified key.

Null if the specified key does not exist or the value is not an eon_cjsonnode object.

2.3.11.2.17 of_valuestring

Description

Gets the string value of the specified key.

Syntax

`of_valuestring` (string *`as_keyname`*)

Argument	Description
`as_keyname`	Name of the key.

Return value

String. The value of the specified key.

Null if the specified key does not exist or the value is not a string.

2.3.11.2.18 Code examples

To add a JSON node with name/value pairs and convert the node to this string:
{"type":"work", "value":"212-555-1234", "pref":"false"}

```
eon_cjsonnode node2
```

```
node2 = Create eon_cjsonnode
node2.of_addkey ("type", "work")
node2.of_addkey ("value", "212-555-1234")
node2.of_addkey ("pref", false)
node2.of_tostring () //exports this node to a string
```

2.3.11.3 eon_cjsonnodearray

2.3.11.3.1 of_append

Description

Adds a value or an eon_cjsonnode object or an eon_cjsonnodearray object at the end of the array.

Syntax 1

`of_append` (any *`aa_keyvalue`*)

Argument	Description
`aa_keyvalue`	Value to be added. Supported data types include long, boolean, string, double, eon_cjsonnode, & eon_cjsonnodearray.

Syntax 2

`of_append` (eon_cjsonnode *`anvo_node`*)

Argument	Description
`anvo_node`	The eon_cjsonnode object to be added.

Syntax 3

`of_append` (eon_cjsonnodearray *`anvo_nodearray`*)

Argument	Description
`anvo_nodearray`	The eon_cjsonnodearray object to be added.

Return value

None

2.3.11.3.2 of_cleararray

Description

Clears all of the values of the current array.

Syntax

`of_cleararray` ( )

Return value

None

2.3.11.3.3 of_getarraynodebyindex

Description

Gets the value (eon_cjsonnodearray object) specified by the index.

Syntax

`of_getarraynodebyindex` (integer *`ai_index`*)

Argument	Description
`ai_index`	Value of the index.

Return value

eon_cjsonnodearray.

Null if the value is not an eon_cjsonnodearray object.

2.3.11.3.4 of_getboolbyindex

Description

Gets the boolean value specified by the index.

Syntax

`of_getboolbyindex` (integer *`ai_index`*)

Argument	Description
`ai_index`	Value of the index.

Return value

Boolean.

Null if the value is not a boolean.

2.3.11.3.5 of_getdoublebyindex

Description

Gets the double value specified by the index.

Syntax

`of_getdoublebyindex` (integer *`ai_index`*)

Argument	Description
`ai_index`	Value of the index.

Return value

Double

Null if the value is not a double.

2.3.11.3.6 of_getlongbyindex

Description

Gets the long value specified by the index.

Syntax

`of_getlongbyindex` (integer *`ai_index`*)

Argument	Description
`astr_connection`	Value of the index.

Return value

Long

Null if the value is not a long.

2.3.11.3.7 of_getnodebyindex

Description

Gets the value (eon_cjsonnode object) specified by the index.

Syntax

`of_getnodebyindex` (integer *`ai_index`*)

Argument	Description
`ai_index`	Value of the index.

Return value

eon_cjsonnode.

Null if the value is not an eon_cjsonnode object.

2.3.11.3.8 of_getstringbyindex

Description

Gets the string value specified by the index.

Syntax

`of_getstringbyindex` (integer *`ai_index`*)

Argument	Description
`ai_index`	Value of the index.

Return value

String

Null if the value is not a string.

2.3.11.3.9 of_gettypebyindex

Description

Gets the data type of the value specified by the index.

Syntax

`of_gettypebyindex` (integer *`ai_index`*)

Argument	Description
`ai_index`	Value of the index.

Return value

Integer. Data type of the value:

0 - null

1 - string

2 - number

3 - boolean (including long and double)

4 - eon_cjsonnodearray

5 - eon_cjsonnode

-1 - Invalid value

2.3.11.3.10 of_insertatindex

Description

Inserts a value or an eon_cjsonnode object or an eon_cjsonnodearray object at the position specified by the index.

Syntax 1

`of_insertatindex` (integer *`ai_index`*, any *`aa_keyvalue`*)

Argument	Description
`ai_index`	Value of the index.
`aa_keyvalue`	The key value to be inserted. Support data types include long, boolean, string, double, eon_cjsonnode, & eon_cjsonnodearray.

Syntax 2

`of_insertatindex` (integer *`ai_index`*, eon_cjsonnode *`anvo_node`*)

Argument	Description
`ai_index`	Value of the index.
`anvo_node`	The eon_cjsonnode object to be inserted.

Syntax 3

`of_insertatindex` (integer *`ai_index`*, eon_cjsonnodearray *`anvo_nodearray`*)

Argument	Description
ai_index	Value of the index.
anvo_nodearray	The eon_cjsonnodearray object to be inserted.

Return value

None

2.3.11.3.11 of_load

Description

Loads the JSON-format string into the current array.

Syntax

`of_load` (string *as_jsonstring*)

Argument	Description
as_jsonstring	The JSON-format string. Make sure the string complies with the JSON and JavaScript language standards.

Return value

Boolean

True - Success.

False - Failed to load.

2.3.11.3.12 of_removebyindex

Description

Removes the value specified by the index.

Syntax

`of_removebyindex` (integer *ai_index*)

Argument	Description
ai_index	Value of the index.

Return value

None

2.3.11.3.13 of_size

Description

Gets the total amount of the values in the array.

Syntax

`of_size` ( )

Return value

Integer. The total amount of values in the array.

-1 - It is called in PowerBuilder, or there is an error.

2.3.11.3.14 of_tostring

Description

Converts the array to the standard JSON-format string.

Syntax

`of_tostring()`

Return value

String. The standard JSON-format string.

2.3.11.3.15 Code examples

To add a JSON array with string values and convert the array to this string: ["displayName","organizations"]

```
eon_cjsonnodearray  array1
array1 = create  eon_cjsonnodearray
array1.of_append ( "displayName" )
array1.of_append ( "organizations" )
array1.of_tostring () //exports this array to a string
```

To add a JSON array with values of various types and convert the array to this string: [{"type":"work", "value":"212-555-1234", "pref":"false"}, {"type":"mobile", "value":"917-555-5432", "pref":"false"}, {"type":"home", "value":"203-555-7890", "pref ":"false"}]

```
eon_cjsonnodearray   array2
array2 = create eon_cjsonnodearray
eon_cjsonnode node2
node2 = Create eon_cjsonnode
node2.of_addkey ("type", "work")
node2.of_addkey ("value", "212-555-1234")
node2.of_addkey ("pref", false)
array2.of_append (node2)
node2.of_setvalue ("type", "mobile")
node2.of_setvalue ("value", "917-555-5432")
node2.of_setvalue ("pref", true)
array2.of_append (node2)
node2.of_setvalue ("type", "home")
node2.of_setvalue ("value", "203-555-7890")
node2.of_setvalue ("pref", false)
array2.of_append (node2)
array2.of_toString() //exports this array to a string
```

2.3.12 Appeon Resize Object

The Appeon resize object helps developers easily resize UI elements to fit with multiple screens, especially for the mobile apps.

2.3.12.1 eon_appeon_resize object

The eon_appeon_resize object provides the most capacity and flexibility because it can not only resize the window and the user object container, but also the controls and the font, at

the same time, it also allows you to dynamically change the resize behavior for a particular window/user object/control at runtime.

2.3.12.1.1 Functions

of_fontresize

Description

Determines whether the font is resized proportionally with the control.

Syntax

`of_fontresize` (boolean *`ab_resize`*, integer *`ai_type`*)

Parameter

`ab_resize` - When true, the font is automatically resized; when false, the font is not resized.

`ai_type` - 1 - Resizes with the height; 2 - Resizes with the width; 3 - Resizes with the height or width whichever has the smaller change in proportion; 4 - Resizes with the height or width whichever has the bigger change in proportion.

Return value

None

of_getscale (internal function)

Description

This is an internal function that will be automatically called when resize script is executed.

Gets the default ratio of each control.

Syntax

`of_getscale` (string *`as_input`*, ref string *`as_flag`*, ref string *`as_xscale`*, ref string *`as_yscale`*, ref string *`as_wscale`*, ref string *`as_hscale`*)

`of_getscale` (string *`as_input`*, ref string *`as_flag`*, ref string *`as_xscale`*, ref string *`as_yscale`*, ref string *`as_wscale`*, ref string *`as_hscale`*, ref string *`as_width`*, ref string *`as_height`*)

`of_getscale` (string *`as_input`*, ref string *`as_flag`*, ref string *`as_xscale`*, ref string *`as_yscale`*, ref string *`as_wscale`*, ref string *`as_hscale`*, ref string *`as_width`*, ref string *`as_height`*, ref string *`as_fontsize`*)

Return value

None

of_gettag

Description

Gets the original tag of a control or object. Original tag includes the information that is not altered by the of_init function.

Syntax

of_gettag (windowobject *awo*)

Parameter

awo - The name of the object or control.

Return value

String. The original tag information of the control or object.

of_init

Description

Initializes the user object container or the window that needs to be resized. You can initialize the user object container or the window only; you can also initialize the controls contained in the user object container or the window by setting the *ab_recursion* parameter to true. Only the initialized user object/window/control can be resized, therefore, this function must be the very first one to be called.

By initializing the user object/window/control, it actually registers the following information to the Tag property of the user object/window/control, separated by "/":

register flag/*resize flag*/*flag*/*x ratio*/*y ratio*/*width ratio*/*height ratio*/*width*/*height*/*x*/*y*/*parentwidth*/*parentheight*/{*textsize*}/*original tag*

flag: used by the developer to manually specify the resize behavior of a control or object, such as scaletobottom, fixtoright etc. By default, the value is "1111". It can be set by using the of_setflag function.

textsize: only effective when the control or object has the textsize property.

Syntax 1: Initializes the user object. If *ab_recursion* is true, all of the controls contained in the user object will be initialized as well.

of_init (userobject *au_uo*, boolean *ab_recursion*)

Syntax 2: Initializes the user object. If *ab_recursion* is true, all of the controls contained in the user object will be initialized as well.

of_init (userobject *au_uo*, integer *ai_width*, integer *ai_height*, boolean *ab_recursion*)

Syntax 3: Initializes the window and the controls in it. If *ab_recursion* is true, all of the user objects and tab controls contained in the window will be initialized as well.

of_init (window *aw_window*, boolean *ab_recursion*)

Parameter

au_uo - The name of the user object container to be initialized.

aw_window - The name of the window container to be initialized, and the controls in the window will be automatically initialized as well.

ai_width - The new width of the user object container. The original proportion of each control will be calculated based on it.

ai_height - The new height of the user object container. The original proportion of each control will be calculated based on it.

ab_recursion - When true, the controls contained in the user object container will be initialized as well; or the user objects and tab controls in the window container will be initialized as well; when false, only the user object container or the window (and controls in the window) will be initialized.

Return value

None

of_init_object

Description

Changes the ratio of a single object or control that has been registered using of_init.

Syntax

of_init_object (windowobject *awo*, decimal *ad_xscale*, decimal *ad_yscale*, decimal *ad_wscale*, decimal *ad_hscale*)

of_init_object (windowobject *awo*, integer *ai_parentwidth*, integer *ai_parentheight*, boolean *ab_recursion*)

of_init_object (windowobject *awo*, boolean *ab_recursion*)

Comparatively speaking, the second syntax is easier to use and produce the desired layout; the first syntax would require more testing to determine an accurate x/y ratio.

The third syntax is for refreshing the object or control easily without needing a lot arguments.

Parameter

awo - The name of the object or control to be initialized.

ad_xscale - The new x ratio.

ad_yscale - The new y ratio.

ad_wscale - The new width ratio.

ad_hscale - The new height ratio.

ai_parentwidth - The new width of the parent container.

ai_parentheight - The new height of the parent container.

ab_recursion - When true, the controls contained in the object or the control will be initialized as well; when false, only the object or the control will be initialized.

Return value

None

of_internal_resize (internal function)

Description

This is an internal function that will be automatically called when resize script is executed.

Resizes the DragObject by utilizing its flag.

Syntax

`of_internal_resize` (ref dragobject *`awo`*, string *`as_flag`*, integer *`ai_newwidth`*, integer *`ai_newheight`*, string *`as_xbl`*, string *`as_ybl`*, string *`as_wbl`*, string *`as_hbl`*)

Parameter

`awo` - The name of the DragObject.

`as_flag` - The flag of the DragObject.

`ai_newwidth` - The new width of the DragObject.

`ai_newheight` - The new height of the DragObject.

`as_xbl` - The x ratio of the DragObject.

`as_ybl` - The y ratio of the DragObject.

`as_wbl` - The width ratio of the DragObject.

`as_hbl` - The height ratio of the DragObject.

Return value

None.

of_orginterval_h (internal function)

Description

This is an internal function that will be automatically called when resize script is executed.

Gets the default height of a control.

Syntax

`of_orginterval_h` (windowobject *`awo`*)

Parameter

`awo` - The name of the object or control.

Return value

Long. The default height of the control.

of_orginterval_w (internal function)

Description

This is an internal function that will be automatically called when resize script is executed.

Gets the default width of a control.

Syntax

`of_orginterval_w` (windowobject *`awo`*)

Parameter

`awo` - The name of the object or control.

Return value

Long. The default width of the control.

of_orginterval_x (internal function)

Description

This is an internal function that will be automatically called when resize script is executed.

Gets the default x coordinate of a control.

Syntax

`of_orginterval_x` (windowobject *awo*)

Parameter

awo - The name of the object or control.

Return value

Long. The default x coordinate of the control.

of_orginterval_y (internal function)

Description

This is an internal function that will be automatically called when resize script is executed.

Gets the default y coordinate of a control.

Syntax

`of_orginterval_y` (windowobject *awo*)

Parameter

awo - The name of the object or control.

Return value

Long. The default y coordinate of the control.

of_pause

Description

Pauses the auto-resizing service for all controls. It is used in pair with of_recover.

Syntax

`of_pause()`

Return value

None.

of_recover

Description

Resumes the auto-resizing service for all controls. It is used in pair with of_pause.

Syntax

`of_recover()`

Return value

None.

of_refresh_object

Description

Refreshes the properties of a single object or control that has been registered using of_init. The object or control's parent will not be refreshed by this function.

Syntax

`of_refresh_object` (dragobject *ado*)

Parameter

ado - The name of the object or control whose size has been changed.

Return value

None.

of_registered (internal function)

Description

This is an internal function that will be automatically called when resize script is executed.

Checks whether a control has been registered/initialized.

Syntax

`of_registered` (windowobject *awo*)

Parameter

awo - The name of the object or control.

Return value

Boolean.

of_resizable (internal function)

Description

This is an internal function that will be automatically called when resize script is executed.

Checks whether a control is resizable.

Syntax

`of_resizable` (windowobject *awo*)

Parameter

awo - The name of the object or control.

Return value

Boolean.

of_resize

Description

Resizes a control according to the registered information.

Syntax 1: Resizes the user object.

of_resize (userobject *au_uo*, integer *ai_newwidth*, integer *ai_newheight*, boolean *ab_recursion*)

Syntax 2: Resizes the window and the controls in the window. If *ab_recursion* is true, objects in the user object and tab controls will be resized as well.

of_resize (window *aw_window*, integer *ai_newwidth*, integer *ai_newheight*, boolean *ab_recursion*)

Syntax 3: Resizes the DataWindow control. Normally the DataWindow control will be resized with its parent container, therefore, it is not recommended to call this function to resize it.

of_resize (datawindow *adw*, integer *ai_newwidth*, integer *ai_newheight*, boolean *ab_zoom*, integer *ai_type*, decimal *ad_maxrate*)

Syntax 4: Resizes the DataWindow control. Normally the DataWindow control will be resized with its parent container, therefore, it is not recommended to call this function to resize it.

of_resize (datawindow *adw*, integer *ai_newwidth*, integer *ai_newheight*, decimal *ad_maxrate*)

Parameter

au_uo - The name of the user object container to be resized.

aw_window - The name of the window to be resized.

adw - The name of the DataWindow to be resized.

ai_newwidth - The new width of the user object container, the window, or the DataWindow.

ai_newheight - The new height of the user object container, the window, or the DataWindow.

ab_zoom - Sets whether the DataWindow control is resizable.

ai_type - The type of resizing. See of_zoom for more.

ad_maxrate - The maximum resize ratio.

ab_recursion - When true, the controls contained in the user object container or the window will be resized as well; when false, only the user object container or the window will be resized.

Return value

None.

of_setflag

Description

Sets the flag of a control or object, so that the control or object will be resized differently.

Syntax

of_setflag (windowobject *awo*, string *as_flag*)

Parameter

awo - The name of the object or control.

as_flag - The flag of the object or control. The value of flag is composed of 4 digits, by default it is "1111".

The first digit indicates the resize behavior of the x coordinate: 0 - Keeps unchanged; 1 - Changes proportionally; 2 - Moves to right horizontally.

The second digit indicates the resize behavior of the y coordinate: 0 - Keeps unchanged; 1 - Changes proportionally; 2 - Moves down vertically.

The third digit indicates the resize behavior of the width: 0 - Keeps unchanged; 1 - Changes proportionally; 2 - Extends to right horizontally; 3 - Extends to left horizontally; 4 - Extends to right edge; 5 - Extends to left edge.

The fourth digit indicates the resize behavior of the height: 0 - Keeps unchanged; 1 - Changes proportionally; 2 - Extends down vertically; 3 - Extends up vertically; 4 - Extends to the bottom; 5 - Extends to the top.

Return value

Boolean.

of_start

Description

Restarts the auto-resizing service for a control or object. It is used in pair with of_stop.

Syntax

`of_start` (windowobject *awo*)

Parameter

awo - The name of the object or control.

Return value

None.

of_stop

Description

Stops the auto-resizing service for a control or object. It is used in pair with of_start.

Syntax

`of_stop` (windowobject *awo*)

Parameter

awo - The name of the object or control.

Return value

None.

of_stringtoarray (internal function)

Description

This is an internal function that will be automatically called when resize script is executed.

Splits a string into an array.

Syntax

`of_stringtoarray` (string *`as_input`*, ref string *`as_array[]`*, string *`as_flag`*)

Return value

None.

of_typeof (internal function)

Description

This is an internal function that will be automatically called when resize script is executed.

Checks the type of a control or an object.

Syntax

`of_typeof` (windowobject *`awo_control`*)

Return value

String. The type of the control or the object.

of_zoom

Description

Determines whether to turn on or off the resizing feature for the DataWindow control.

Syntax

`of_zoom` (boolean *`ab_zoom`*, integer *`ai_type`*)

`of_zoom` (boolean *`ab_zoom`*, integer *`ai_type`*, decimal *`adec_rate`*)

Parameter

`ab_zoom` - When true, resizing feature for the DataWindow control is enabled; when false, resizing feature is disabled.

`ai_type` - 1 - Resizes proportionally with height; 2 - Resizes proportionally with width; 3 - Resizes with height or width whichever has the smaller change in proportion; 4: Resizes with height or width whichever has the bigger change in proportion.

`adec_rate` - The maximum resizing rate of the DataWindow control. By default, it is 3, which means the DataWindow control can be resized by no more than 300%.

Return value

None.

2.3.12.1.2 Code example

Below is the sample code for resizing a user object. The code for resizing a window is similar.

```
//declears the instance variable for the user object
eon_appeon_resize  ieon_resize
```

```
//initializes in the constructor event of the user object
//(if it is window, initializes in the open event of the window)
ieon_resize = create eon_appeon_resize
// registers all of the controls contained in this user object
ieon_resize.of_init( this, true)
//enables font resizing and resizes with the width
ieon_resize.of_fontresize( true, 2)
//enables datawindow resizing and resizes with the width
ieon_resize.of_zoom( true, 2)
//changes the datawindow resize behavior by keeping x and y coordinates unchanged,
//and scaling to the right edge and to the bottom of the user object container.
ieon_resize.of_setflag( dw_1, '0044')
//keeps the cb_1 size unchanged, and also keeps its relative position unchanged
//which means the distances from the right edge and from the bottom remain the
 same.
ieon_resize.of_setflag( cb_1, '2222')

//defines the resize event for the user object, event ID is set to pbm_size
//Resizes all of the controls contained in the user object
ieon_resize.of_resize( this, newwidth, newheight, true)

//destroys the user object in the destructor event
Destroy  ieon_resize
```

2.4 Mobile Device API

The APIs listed in this section are specific for the mobile environment including iOS and Android. These APIs allow developers to take advantage of the iOS SDK APIs and Android SDK APIs to access the native features of the target mobile device.

Note: For each category of APIs, there are two objects with almost the same name -- one with the "ex" suffix and the other without the suffix, e.g. eon_mobile_awsex and eon_mobile_aws objects. The former is the extension of the latter, and you are recommended to directly use the object with the "ex" suffix.

2.4.1 Audio

Plays the audio file using the device's audio player.

If more than one audio object instance is created and their of_play function is called, then more than one audio file is being played at the same time. This could happen if the window is closed without destroying the audio instance (so the audio is still being played), and then when the window is opened again, the audio instance is created again.

Supported on iOS device only.

2.4.1.1 eon_mobile_audioex

2.4.1.1.1 Properties

integer ii_errorcode

Error code which is returned when the oe_error event is triggered.

powerobject ipo_bindevent

The object to bind with the oe_error event.

string is_bindwitherrorevent

The name of the event bound with the oe_error event of the powerobject ipo_bindevent.

string is_errortext

Error message which is returned when the oe_error event is triggered.

2.4.1.1.2 Events

oe_error

Description

It will be triggered automatically if there is an error when using the audio functionality.

Supported on iOS device only.

Note: This event will update the value of is_errortext and ii_errorcode properties.

Syntax

None.

Return value

None.

2.4.1.1.3 Functions

of_getstatus

Description

Gets the current status of the audio file.

Supported on iOS device only.

Syntax

```
audio.of_getstatus()
```

Argument Type	Argument	Description
eon_mobile_audioex	*audio*	The name of the eon_mobile_audioex object.

Return value

Integer.

0 - The audio file is stopped.

1 - The audio file is being played.

2 - The audio file is paused.

-1 - It is called in PowerBuilder, PowerServer Web, or Android, or there is an error.

of_pauseorresume

Description

Pauses the audio file that is being played or resumes playing it from where it stopped.

Supported on iOS device only.

Syntax

audio.of_pauseorresume ( )

Argument Type	Argument	Description
eon_mobile_audioex	*audio*	The name of the eon_mobile_audioex object.

Return value

Integer.

1 - The audio file is paused successfully.

-1 - It is called in PowerBuilder, PowerServer Web, or Android, or there is an error.

of_play

Description

Uses the default audio player application to play the audio file.

The supported audio file formats are .aac, .mp3, .aiff, and .wav. Not all the audio files with these suffixes can be played; if some media files cannot be played, they should be transcoded first via iTunes.

Supported on iOS device only.

Syntax

audio.of_play (string *as_filepath*)

Argument Type	Argument	Description
eon_mobile_audioex	*audio*	The name of the eon_mobile_audioex object.
string	*as_filepath*	The full path of the audio file to be played.

Return value

Integer.

1 - The audio file is played successfully.

-1 - It is called in PowerBuilder, PowerServer Web, or Android, or there is an error.

of_register

Description

Registers the object and the event to be bound with the oe_error event.

Supported on iOS device only.

Syntax

audio.of_register (powerobject *apb_bind*, string *as_error*)

Argument Type	Argument	Description
eon_mobile_audioex	*audio*	The name of the eon_mobile_audioex object.
powerobject	*apb_bind*	The object to be bound with the oe_error event.
string	*as_error*	The event to be bound with the oe_error event.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder, PowerServer Web, or Android, or there is an error.

of_stop

Description

Stops the audio file that is being played.

Supported on iOS device only.

Syntax

audio.of_stop ( )

Argument Type	Argument	Description
eon_mobile_audioex	*audio*	The name of the eon_mobile_audioex object.

Return value

Integer.

1 - The audio file is stopped successfully.

-1 - It is called in PowerBuilder, PowerServer Web, or Android, or there is an error.

of_triggerevent

Description

Triggers the object event bound with the oe_error event.

Supported on iOS device only.

Syntax

audio.of_triggerevent (string *as_event*)

Argument Type	Argument	Description
eon_mobile_audioex	*audio*	The name of the eon_mobile_audioex object.

Argument Type	Argument	Description
string	*as_event*	The event name bound with oe_error event.

Return value

None

2.4.2 Appeon Workspace

Manipulates the various features of the Appeon mobile application and the Appeon Workspace, such as the screen orientation, title bar, assistive touch bar, log file, application information, PowerServer Mobile version, etc.

Note: The Appeon Workspace API takes effect in both the mobile app running in Appeon Workspace and the native standalone mobile applications.

2.4.2.1 eon_mobile_awsex object

2.4.2.1.1 Properties

powerobject ipo_bindevent

PowerBuilder object bound with the oe_urlschemesucceed event.

string is_bindwithsucceedevent

The name of the event bound with the oe_urlschemesucceed event.

2.4.2.1.2 Events

oe_urlschemesucceed

Description

This event is triggered when Appeon Workspace is launched by other Appeon mobile apps or third-party apps via the URL scheme.

This event can be bound to an object and event using the of_register and of_triggerevent function.

Supported on mobile client only.

Syntax

aws.oe_urlschemesucceed (string *as_urlscheme*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
string	*as_urlscheme*	Receives the value of the URL scheme parameter passed by the other Appeon mobile app or third-party app.

Return value

None.

2.4.2.1.3 Functions

of_checkpermissionfornet

Description

Checks if the app in Appeon Workspace has permissions to access the .NET IIS server.

Supported on mobile client only.

Syntax

aws.of_checkpermissionfornet (string *as_workspaceid*, string *as_appname*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
string	*as_workspaceid*	ID of Appeon Workspace.
string	*as_appname*	Name of the application installed in Appeon Workspace.

Return value

Integer.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

-2 - Server is not .NET IIS server.

-3 - Failed to get the application name from Appeon Workspace.

-6 - Failed to execute AppeonDotNetComponent.of_ExecInterface.

1 - The app has permissions to access the server.

20 - Appeon Workspace ID does not exist.

10 - Appeon Workspace ID exists, but is disabled.

11 - Appeon Workspace ID and the app is not associated.

30 - Other reasons.

of_clearlog

Description

Clears the mobile app log information.

Supported on mobile client only.

Syntax

aws.of_clearlog ()

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getappinfo

Description

Gets the information (such as app name, app URL) of the current Appeon mobile application.

Supported on mobile client only.

Syntax

aws.of_getappinfo (ref string *as_appname*, ref string *as_appurl*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
string	*as_appname*	Returns the name of the current Appeon mobile application. Returns empty string if it is called in PowerBuilder or PowerServer Web or if there is any error.
string	*as_appurl*	Returns the application URL that is set in the "App URL" field of the current Appeon mobile application. Returns empty string if it is called in PowerBuilder or PowerServer Web or if there is any error.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getapporientation

Description

Detects if the screen orientation of the Appeon mobile application is Landscape or Portrait.

Supported on mobile client only.

This function is only valid for the current running application.

Syntax

```
aws.of_getapporientation ()
```

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.

Return value

Integer.

0 - Unknown.

1 - Portrait.

2 - Landscape.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getapprotationlock

Description

Detects if the screen rotation of the Appeon mobile application is locked.

Supported on mobile client only.

Syntax

```
aws.of_getapprotationlock ()
```

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.

Return value

Integer.

1 - Locked.

0 - Unlocked.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getassistivetouchbtnvisible

Description

Detects if the assistive touch bar is visible.

Supported on mobile client only.

Syntax

```
aws.of_getassistivetouchbtnvisible ()
```

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.

Return value

Integer.

1 - The assistive touch bar is visible.

0 - The assistive touch bar is invisible.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getassistivetouchmode

Description

Gets the current assistive touch mode. For more about assistive touch mode, refer to Section 2.2, “Event-handling model” in *Mobile UI Design & Development Guide*.

Supported on mobile client only.

Syntax

```
aws.of_getassistivetouchmode ()
```

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.

Return value

Integer.

0 - The assistive touch mode is Left-click.

1 - The assistive touch mode is Right-click.

2 - The assistive touch mode is Drag.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getcloseappiconvisible

Description

Detects if the close app icon is visible or not. For more about the close app icon, refer to the section called “Window title bar, menu, & toolbar” in *Mobile UI Design & Development Guide*.

Supported on mobile client only.

Syntax

aws.of_getcloseappiconvisible ()

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.

Return value

Integer.

1 - The close app icon is visible.

0 - The close app icon is invisible.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getdwinputmode

Description

Obtains the display mode of the virtual keyboard on the mobile device when the user taps into the DataWindow text field. If of_setdwinputmode is called, then of_getdwinputmode will obtain the display mode set via of_setdwinputmode. If of_setdwinputmode is not called, then of_getdwinputmode will obtain the display mode setting in Appeon Workspace.

Supported on mobile client only.

Syntax

aws.of_getdwinputmode ()

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.

Return value

Integer.

1 - Keyboard displays when the user taps once in the DataWindow field.

2 - Keyboard displays when the user taps twice in the DataWindow field.

-1 - It is called in PowerBuilder or Appeon Web, or there is an error.

See also

of_setdwinputmode

of_getdwmousemovemode

Description

Detects if the pbm_dwnmousemove event ID of DataWindow is supported.

Supported on mobile client only.

Syntax

aws.of_getdwmousemovemode ()

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.

Return value

Integer.

1 - Supported.

0 - Unsupported.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_gettitlebarvisible

Description

Detects if the application title bar is visible or invisible. For more about the titlebar, refer to the section called “Window title bar, menu, & toolbar” in *Mobile UI Design & Development Guide*.

Supported on mobile client only.

Syntax

aws.of_gettitlebarvisible ( )

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.

Return value

Integer.

1 - The title bar is visible.

0 - The title bar is invisible but the normal view icon is visible.

2 - Both the title bar and the normal view icon are invisible.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_geturlschemeparm

Description

Gets the value of the URL scheme parameter passed by another Appeon mobile app or a third-party app.

For more about the URL scheme, refer to URL Scheme Name in *PowerServer Toolkit User Guide*.

Supported on mobile client only.

Syntax

aws.of_geturlschemeparm (ref string *as_urlscheme*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
ref string	*as_urlscheme*	Receives the value of the URL scheme parameter passed by the third-party app.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getversion

Description

Gets the version number of Appeon Workspace.

Supported on mobile client only.

Syntax

aws.of_getversion (ref string *as_version*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
string	*as_version*	Returns the Appeon Workspace version number. Returns empty string if it is called in PowerBuilder or PowerServer Web, or if there is any error.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getwindowlisticonvisible

Description

Detects if the window list icon is visible or not. For more about the window list icon, refer to the section called “Window title bar, menu, & toolbar” in *Mobile UI Design & Development Guide*.

Supported on mobile client only.

Syntax

aws.of_getwindowlisticonvisible()

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.

Return value

Integer.

1 - The window list icon is visible.

0 - The window list icon is invisible.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getworkspaceheight

Description

Gets the height (in PBU) of the Appeon Workspace screen. If the workspace title bar is visible, then its height is not included.

Supported on mobile client only.

Syntax

aws.of_getworkspaceheight ()

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.

Return value

Integer

The height of the Appeon Workspace screen.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getworkspaceidappname

Description

Gets the Appeon Workspace ID and the name of the application currently running in Appeon Workspace. The Appeon Workspace ID is based on the device's unique identifier which depends on the device manufacturer, product type, and operating system.

Supported on mobile client only.

Syntax

aws.of_getworkspaceidappname ()

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.

Argument Type	Argument	Description
string	*as_workspaceid*	The Appeon Workspace ID.
string	*as_appname*	The name of the application currently running in Appeon Workspace.

Return value

Integer

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getworkspacewidth

Description

Gets the width (in PBU) of the Appeon Workspace screen.

Supported on mobile client only.

Syntax

aws.of_getworkspacewidth ( )

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.

Return value

Integer

The width of the Appeon Workspace screen.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_log

Description

Writes the log information to the mobile client log, and the default log level is INFO.

Supported on mobile client only.

Syntax

aws.of_log (value integer *ai_level*, value string *as_info*)

aws.of_log (value string *as_info*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
integer	*ai_level*	1 - FATAL level. 2 - ERROR level.

Argument Type	Argument	Description
		3 - WARNING level. 4 - INFO level. (Default value) 5 - DEBUG level.
string	*as_info*	The log information.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

Usage

For the log path of mobile applications, please refer to Section 5.4.6.8, “Client Logs” in *PowerServer Configuration Guide for .NET* or PowerServer Configuration Guide for J2EE.

of_register

Description

Registers the object and the event to be bound with the oe_urlschemesucceed event.

Supported on mobile client only.

Syntax

aws.of_register (powerobject apb_bind, string *as_schemeparam*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
powerobject	*apb_bind*	The object to be bound with the oe_urlschemesucceed event.
string	*as_schemeparam*	The event to be bound with the oe_urlschemesucceed event.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_sendmail

Description

Sends an email from the mobile application.

Supported on mobile client only.

Syntax

aws.of_sendmail (value eon_mobile_str_mailcontent *astr_content*, value string *as_recipient[]*, value string *as_cc[]*, value string *as_bcc[]*, value string *as_attachmentfile[]*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
eon_mobile_str_mailcontent	*astr_content*	The structure of the email content. See variable list of eon_mobile_str_mailcontent.
string	*as_recipient[]*	Specifies the recipient list of the mail.
string	*as_cc[]*	Specifies the Cc recipient list of the mail.
string	*as_bcc[]*	Specifies the Bcc recipient list of the mail.
string	*as_attachmentfile[]*	Attachment file path list.

Return value

Integer. The following return values are effective for the iOS device only. On the Android device, 0 will be returned under all circumstances, because the corresponding mail API from the Android SDK returns no standardized value.

1 - Send the email successfully.

0 - Cancel sending the email.

-1 - Failed to send the email, or it is called in PowerBuilder or PowerServer Web, or there is an error.

-2 - The email account is not configured.

-100 - System error.

Code example

Refer to How to generate a PDF file and send it via email for the code example of generating and sending a PDF file via email.

of_setapporientation

Description

Sets the screen orientation of the Appeon mobile application.

The screen orientation is set regardless of the current screen rotation.

After this function is called successfully to set the screen orientation, be sure to call of_setapprotationlock to lock the screen orientation. And do not call of_setapprotationlock

before this function is called. If of_setapprotationlock is called before this function is called, this function will have no effect and return -1.

Supported on mobile client only.

Syntax

aws.of_setapporientation (value integer *ai_mode*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
integer	*ai_mode*	1 - Sets screen orientation to Portrait. 2 - Sets screen orientation to Landscape.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_setapprotationlock

Description

Sets whether to lock the screen orientation of the Appeon mobile application, so the screen orientation will not change with the screen rotation of the device.

This function must be called after of_setapporientation is called successfully to set the screen orientation.

Supported on mobile client only.

Syntax

aws.of_setapprotationlock (value integer *ai_mode*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
integer	*ai_mode*	1 - Locks the screen orientation of Appeon mobile application, so it will not rotate when the screen orientation of the device changes. 0 - Unlocks the screen orientation of the Appeon mobile application.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_setassistivetouchbtnvisible

Description

Sets whether the assistive touch bar is visible.

Supported on mobile client only.

Syntax

aws.of_setassistivetouchbtnvisible (value integer *ai_mode*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
integer	*ai_mode*	1 - Sets the assistive touch bar to be visible. 0 - Sets the assistive touch bar to be invisible.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_setassistivetouchmode

Description

Sets the assistive touch mode.

Supported on mobile client only.

Syntax

aws.of_setassistivetouchmode (value integer *ai_mode*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
integer	*ai_mode*	0 - Sets to the Left-click mode. 1 - Sets to the Right-click mode.

Argument Type	Argument	Description
		2 - Sets to the Drag mode.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_setcloseappiconvisible

Description

Sets whether the close app icon is visible or not.

Supported on mobile client only.

Syntax

aws.of_setcloseappiconvisible (value integer *ai_mode*, value string *as_title*, value string *as_message*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
integer	*ai_mode*	1 - Sets the close app icon to be visible. 0 - Sets the close app icon to be invisible.
integer	*as_title*	The title that displays in the dialog box which pops up when the Close App icon is tapped.
string	*as_message*	The message that displays in the dialog box which pops up when the Close App icon is tapped. A **Yes**/**No** button will be displayed in the dialog box. When Yes is tapped, the app will be closed. When No is tapped, the app will not be closed and will continue running. If *as_message* is null or blank, then no message box will be displayed and the app will be closed immediately.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_setdwinputmode

Description

Sets the display mode of the virtual keyboard on the mobile device when the user taps into the DataWindow text field. You can also set the display mode for all applications installed in the same Appeon Workspace, by going to Appeon Workspace | **Settings** | **General**, and setting the **Show Keyboard on a Single Tap** option. For more information, refer to Section 5.5, "Showing keyboard on single tap or double taps" in *Appeon Workspace User Guide*. The of_setdwinputmode function takes precedence over the setting in Appeon Workspace.

By default, the keyboard pops up when the user taps twice in the field of the DataWindow, or when the user presses the DataWindow field for as long as 1500 milliseconds (you can change this value using of_setlongpressinterval).

Supported on mobile client only.

Syntax

aws.of_setdwinputmode (integer *ai_mode*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
integer	*ai_mode*	1 – Keyboard displays when the user taps once on a DataWindow field. 2 – Keyboard displays when the user taps twice on a DataWindow field.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

Limitation

When the user is dragging in the DataWindow control, the virtual keyboard may display unexpectedly.

When the virtual keyboard displays after the single tap, the DataWindow text field will not respond to the scroll gesture any more.

See also

of_getdwinputmode

of_setdwmousemovemode

Description

Sets whether to support the pbm_dwnmousemove event ID of DataWindow.

Supported on mobile client only.

Syntax

aws.of_setdwmousemovemode (value integer *ai_mode*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
integer	*ai_mode*	1 - Supported. 0 - Unsupported.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_sethinttext

Description

Sets the hint text for the SingleLineEdit control. The hint text property is not available in PowerBuilder SingleLineEdit control. It is specially designed for the Appeon mobile application.

Syntax

aws.of_sethinttext (singlelineedit *sle_target*, string *as_hinttext*)

Argument	Description
aws	The name of the eon_mobile_awsex object.
sle_target	The name of the eon_singlelineeditex object.
as_hinttext	The text that will be displayed as hint text in the SingleLineEdit control.

Return value

Integer.	1 - Success. -1 - It is called in PowerBuilder or PowerServer Web, or there is an error. -2 - The *as_hinttext* argument is null.

of_sethinttextcolor

Description

Sets the color of the hint text for the SingleLineEdit control. The hint text color property is not available in PowerBuilder SingleLineEdit control. It is specially designed for the Appeon mobile application.

Syntax

`aws.of_sethinttextcolor` (singlelineedit *`sle_target`*, long *`al_newcolor`*)

Argument	Description
`aws`	The name of the eon_mobile_awsex object.
`sle_target`	The name of the eon_singlelineeditex object.
`al_newcolor`	The color of the hint text that will be displayed in the SingleLineEdit control.

Return value

Integer.	1 - Success. -1 - It is called in PowerBuilder or PowerServer Web, or there is an error. -2 - The *`al_newcolor`* argument is null.

of_setlongpressinterval

Description

Sets the duration (in milliseconds) for a press before it is recognized as a long press. Once this duration is reached, the keyboard will be brought up to display. By default, the keyboard pops up when the user taps twice in the field of the DataWindow, or when the user presses the DataWindow field for as long as 1500 milliseconds.

Supported on mobile client only.

Syntax

`aws``.of_setlongpressinterval` (long *`al_interval`*)

Argument Type	Argument	Description
eon_mobile_awsex	*`aws`*	The name of the eon_mobile_awsex object.
long	*`al_interval`*	Sets the duration in milliseconds. If it is set to a value smaller than 500 milliseconds, then 550 milliseconds will be used, because 500 milliseconds is already used as the duration for the DataWindow drag & drop event.

Return value

Long.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_settitlebarvisible

Description

Sets whether the application title bar is visible or invisible. When the title bar is set to invisible, the application will be displayed in full screen view.

Supported on mobile client only.

Syntax

aws.of_settitlebarvisible (value integer *ai_mode*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
integer	*ai_mode*	1 - Sets the title bar to be visible. 0 - Sets the title bar to be invisible, and displays the normal view icon on the top right corner of the window, when it is tapped, the app will return to the normal view (with the title bar visible again). 2 - Sets the title bar to be invisible, and displays no icon for showing the title bar again.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_setwindowlisticonvisible

Description

Sets whether the window list icon is visible or not.

Supported on mobile client only.

Syntax

aws.of_setwindowlisticonvisible (value integer *ai_mode*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
integer	*ai_mode*	1 - Sets the window list icon to be visible. 0 - Sets the window list icon to be invisible.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_triggerevent

Description

Triggers the object event bound with the oe_urlschemesucceed event.

Supported on mobile client only.

Syntax

aws.of_triggerevent (string *as_event*)

Argument Type	Argument	Description
eon_mobile_awsex	*aws*	The name of the eon_mobile_awsex object.
string	*as_event*	The event name bound with oe_urlschemesucceed event.

Return value

None.

2.4.2.1.4 Structures

eon_mobile_str_mailcontent

Description

Struct.

The content information of an email.

Property

Type	Variable Name	Description
string	s_subject	Specifies the subject of the mail.

Type	Variable Name	Description
string	s_notetext	Specifies the content of the mail body.
boolean	b_html	Sets if the mail format is HTML.

2.4.3 Barcode

Creates or reads the information of a barcode.

Usage example

- Archives administration:

  Facilitate the separation and indexing of documents that have been imaged in batch scanning applications.

- Healthcare industry:

  Patient identification (to access patient data, including medical history, drug allergies, etc.)

- Pharmaceutical industry:

  Identify, secure and track products to deal with competitors who want to market counterfeit or fake goods.

- Retail industry:

  Help track items and also reduce instances of shoplifting involving price tag swapping.

- Tag reader:

  Use barcodes to connect customers from your offline marketing materials to information, entertainment, and interactive experiences on their smartphones.

2.4.3.1 eon_mobile_barcodeex object

2.4.3.1.1 Functions

of_create

Description

Generates an image containing the barcode according to the data and format specified by the user, and returns the image path.

Supported on mobile client only.

Syntax

barcode.of_create (value string *as_data*, value integer *ai_format*, ref string *as_filepath*)

barcode.of_create (value string *as_data*, value integer *ai_format*, ref string *as_filepath*, ref blob *ablb_data*)

Argument Type	Argument	Description
eon_mobile_barcodeex	*barcode*	The name of the eon_mobile_barcodeex object.
string	*as_data*	Sets the data of a barcode.
integer	*ai_format*	Sets the format of barcode.
string	*as_filepath*	Returns the path of the generated barcode image in the application sandbox's "plugin" directory.
blob	*ablb_data*	Returns the data of the generated barcode image.

Following is the list of supported barcode formats:

Data	Format	Specification
0	Unknown	
3	EAN-8	It is the short version of EAN-13. An EAN-8 number contains 7 digits of message plus 1 check digit. The first two or three digits identify the numbering authority, and the remaining 4 or 5 digits identify the product.
4	UPC-E4	It is the short form representation of a UPC-A number. It reduces the data length from 12 digits to 6 digits by compressing the extra zeros. It is suited for identifying products in small packages.
6	UPC-A	It is the most common and well-known symbology in the US, which consists of four areas: 1) The number system; 2) The manufacturer code; 3) The product code; 4) The check digit.
7	EAN-13	It is a superset of UPC-A. Any software or hardware capable of reading an EAN-13 symbol will automatically be able to read an UPC-A symbol.
11	CODE 128	A high-density linear symbology that encodes text, numbers, numerous functions and the entire 128 ASCII character set. It enables you to store diversified information in the barcode.
12	CODE 93	An alphanumeric, variable length symbology designed to provide a higher density and data security enhancement to Code 39.
13	CODE 39	The first alpha-numeric symbology that originally encoded 39 characters (now 43) . It is not as compact as the Code 128 symbology, but it is still

Data	Format	Specification
		heavily used in the automotive industry and by the US Department of Defense.
14	ITF	A mix between the POS (Point-of-sale) codes and the logistics barcode Code 128. It's mainly being used in trading, but only on products that need no POS interaction.
20	QR CODE	A "2D Matrix" symbology. Because QR Code requires camera based scanners it is currently restricted for use with applications that will involve imaging scanners within mobile devices and not for POS processing.
21	DataMatrix	A 2D barcode symbology with very high data density. It is spreading in the area of mobile marketing, in such applications the DataMatrix barcode is also known under the name SemaCode.
22	AZTEC	Supported on iOS only. It is a matrix symbology which supports the entire ASCII character set and offers several error checking modes. Aztec symbols are square and may be read at any orientation.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getfiledata

Description

Reads data from the specified image file.

Syntax

barcode.of_getfiledata (string *as_filepath*)

Argument Type	Argument	Description
eon_mobile_barcodeex	*barcode*	The name of the eon_mobile_barcodeex object.
string	*as_filepath*	The full path of the file you want to read.

Return value

Blob.

of_read

Description

Executes this function when you need to scan an image containing the barcode information.

The camera will be opened automatically by Appeon, and you need to point the camera at the image to be scanned. The camera will automatically close after a successful scanning, and will return the data and format contained in the barcode.

Supported on mobile client only.

Syntax

barcode.of_read (ref string *as_data*, ref integer *ai_format*)

Argument Type	Argument	Description
eon_mobile_barcodeex	*barcode*	The name of the eon_mobile_barcodeex object.
string	*as_data*	Returns the data of a barcode.
integer	*ai_format*	Returns the format of barcode.

Following is the list of supported barcode formats:

Data	Format	Specification
0	Unknown	
3	EAN-8	It is the short version of EAN-13. An EAN-8 number contains 7 digits of message plus 1 check digit. The first two or three digits identify the numbering authority, and the remaining 4 or 5 digits identify the product.
4	UPC-E4	It is the short form representation of a UPC-A number. It reduces the data length from 12 digits to 6 digits by compressing the extra zeros. It is suited for identifying products in small packages.
6	UPC-A	It is the most common and well-known symbology in the US, which consists of four areas: 1) The number system; 2) The manufacturer code; 3) The product code; 4) The check digit.
7	EAN-13	It is a superset of UPC-A. Any software or hardware capable of reading an EAN-13 symbol will automatically be able to read an UPC-A symbol.
11	CODE 128	A high-density linear symbology that encodes text, numbers, numerous functions and the entire 128 ASCII character set. It enables you to store diversified information in the barcode.

Data	Format	Specification
12	CODE 93	An alphanumeric, variable length symbology designed to provide a higher density and data security enhancement to Code 39.
13	CODE 39	The first alpha-numeric symbology that originally encoded 39 characters (now 43) . It is not as compact as the Code 128 symbology, but it is still heavily used in the automotive industry and by the US Department of Defense.
14	ITF	A mix between the POS (Point-of-sale) codes and the logistics barcode Code 128. It's mainly being used in trading, but only on products that need no POS interaction.
20	QR CODE	A "2D Matrix" symbology. Because QR Code requires camera based scanners it is currently restricted for use with applications that will involve imaging scanners within mobile devices and not for POS processing.
21	DataMatrix	A 2D barcode symbology with very high data density. It is spreading in the area of mobile marketing, in such applications the DataMatrix barcode is also known under the name SemaCode.
22	AZTEC	Supported on iOS only. It is a matrix symbology which supports the entire ASCII character set and offers several error checking modes. Aztec symbols are square and may be read at any orientation.

Return value

Integer.

1 - Success.

0 - Cancel.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

2.4.4 Camera

Captures or views a photo or video using the device's camera.

2.4.4.1 eon_mobile_cameraex object

2.4.4.1.1 Functions

of_getfiledata

Description

Reads data from the specified image file. If the file is over 4 MB, calling this function may cause memory issue in PB 9 or earlier version. You can upgrade your source code to PB 10 or later, and then call the function of the Appeon Workarounds PBL for the corresponding PB version.

Syntax

camera.of_getfiledata (value string *as_filepath*)

Argument Type	Argument	Description
eon_mobile_cameraex	*camera*	The name of the eon_mobile_cameraex object.
string	*as_filepath*	The full path of the file you want to read.

Return value

Blob.

of_openalbums

Description

Opens the album for the user to select a photo or video.

Supported on mobile client only.

Syntax

camera.of_openalbums (ref string *as_filepath*, ref blob *ablb_data*)

camera.of_openalbums (ref string *as_filepath*)

Argument Type	Argument	Description
eon_mobile_cameraex	*camera*	The name of the eon_mobile_cameraex object.
string	*as_filepath*	In iOS, it returns the full path pointing to the selected file in the sandbox plugin directory. The selected video or photo file will be copied from the album to the plugin directory of the current application's sandbox. The files in this plugin directory will be removed when the application exits. In Android, it returns the full path pointing to the selected file.
blob	*ablb_data*	Returns the file data of the photo file or video file.

Argument Type	Argument	Description
		The file data could be saved into the database via the UpdateBlob SQL statement.

Return value

Integer.

1 - Select a photo file successfully, and *as_filepath* will return the full path pointing to the selected photo file.

2 - Select a video file successfully, and *as_filepath* will return the full path pointing to the selected video file.

0 - Cancel selecting a file from the album.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

Code example 1

```
eon_mobile_cameraex ln_camera
String ls_filepath
Integer li_filetype
Integer li_return

li_filetype = 2 // Simultaneously displays the photo or video in the album
ln_camera = create eon_mobile_cameraex
li_return = ln_camera.of_openalbums (ls_filepath)
```

Code example 2

```
eon_mobile_cameraex ln_camera
String ls_filepath
Integer li_filetype
Integer li_return
Blob lb_data

li_filetype = 2 // Simultaneously displays the photo or video in the album
ln_camera = create eon_mobile_cameraex
li_return = ln_camera.of_openalbums (ls_filepath, lb_data)
```

of_takefile

Description

Opens the default camera application and uses it to take a photo or record a video.

Once the photo or video is taken, the camera application automatically closes with all the files saved in the album, and returns to the current application.

The supported video file format is MOV, and the supported photo file format is JPG.

Supported on mobile client only.

Syntax

camera.of_takefile (value integer *ai_filetype*, value boolean *ab_allowedit*, ref string *as_filepath*)

camera.of_takefile (value integer *ai_filetype*, value boolean *ab_allowedit*, ref string *as_filepath*, ref blob *ablb_data*)

camera.of_takefile (value eon_mobile_str_cameraoption *astr_option*, ref string *as_filepath*)

camera.of_takefile (value eon_mobile_str_cameraoption *astr_option*, ref string *as_filepath*, ref blob *ablb_data*)

Argument Type	Argument	Description
eon_mobile_cameraex	*camera*	The name of the eon_mobile_cameraex object.
eon_mobile_str_cameraoption	*astr_option*	Sets the camera parameters. See variable list of eon_mobile_str_cameraoption.
integer	*ai_filetype*	Sets the type of file to take. The camera application will be set to this type by default when it is opened. Users can also manually switch the type in the camera application. 1 - Takes a photo. 2 or any other integer - Records a video.
boolean	*ab_allowedit*	Sets if the file is editable after picture-taking or video-recording.
string	*as_filepath*	Returns the full path pointing to the file in the "plugin" directory. In iOS, the video and photo files will be saved to the iOS album, as well as to the "plugin" directory of the current application's sandbox. As there is restricted access to the directory of iOS album, only the "plugin" directory can be obtained and returned. The files in this "plugin" directory will be removed when the application exits. In Android, the video and photo files will be only saved to the "plugin" directory of the current application's sandbox. The files in the

Argument Type	Argument	Description
		"plugin" directory will remain when the application exits and will be displayed when the Android album is opened by of_openalbums.
blob	*ablb_data*	Returns the file data of the photo file or video file. The file data could be saved into the database via the UpdateBlob SQL statement.

Return value

Integer.

1 - Take a photo successfully, and *as_filepath* will return the full path of the photo file, and return empty string if the picture-taking is cancelled, or if there is any error.

2 - Record a video successfully, and *as_filepath* will return the full path of the video file, and return empty string if the video-recording is cancelled, or if there is any error.

0 - Cancel taking a photo or making a video.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

Code example 1

```
eon_mobile_str_cameraoption lstr_option
eon_mobile_cameraex ln_camera
String ls_filepath
Integer li_return

lstr_option.i_filetype = 1 // Takes the photo
lstr_option.b_allowedit = true // The photo is editable after picture-taking

ln_camera = create eon_mobile_cameraex
li_return = ln_camera.of_takefile (lstr_option, ls_filepath)
```

Code example 2

```
eon_mobile_str_cameraoption lstr_option
eon_mobile_cameraex ln_camera
String ls_filepath
Integer li_return
blob lb_data
lstr_option.i_filetype = 1 // Takes the photo
lstr_option.b_allowedit = true // The photo is editable after picture-taking

ln_camera = create eon_mobile_cameraex
li_return = ln_camera.of_takefile (lstr_option, ls_filepath, lb_data)
```

2.4.4.1.2 Structures

eon_mobile_str_cameraoption

Description

Struct.

Optional parameters to customize the camera settings.

Property

Type	Variable Name	Description
integer	`i_filetype`	Sets the type of file to take. The camera application will be set to this type by default when it is opened. Users can also manually switch the type in the camera application. 1 - Takes a photo. 2 - Records a video.
boolean	`b_allowedit`	Sets if the file is editable after the picture is taken or the video is recorded.

2.4.5 Connection

Obtains the network state and connection type.

2.4.5.1 eon_mobile_connectionex object

2.4.5.1.1 Functions

of_getconnectioninfo

Description

Gets the device's network connection information.

Supported on mobile client only.

Syntax

`connection``.of_getconnectioninfo ()`

Argument Type	Argument	Description
eon_mobile_connectionex	*`connection`*	The name of the eon_mobile_connectionex object.

Return value

Integer.

0 - No connection.

1 - Cell network: 2G/3G/4G.

2 - Wi-Fi network.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

2.4.6 Device

Obtains the device specific information, such as the device type, the device DPI/PPI, the OS version, the device's memory, the device's OS, the screen resolution, etc.

2.4.6.1 eon_mobile_deviceex object

2.4.6.1.1 Functions

of_getdeviceid

Description

Gets the device's unique identifier, which depends on the device manufacturer, product type, and operating system.

Supported on mobile client only.

Syntax

device.of_getdeviceid (ref string *as_id*)

Argument Type	Argument	Description
eon_mobile_deviceex	*device*	The name of the eon_mobile_deviceex object.
string	*as_id*	Returns the device unique identifier. Returns empty string if it is called in PowerBuilder or PowerServer Web or, if there is any error.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

Usage

Use the of_getdeviceid() function to allow the application to run on specific mobile devices that are authorized. In this way, if unauthorized users install the app on their own mobile device and somehow steals a valid username/password they still will not be able to run the app or your data.

of_getdevicetype

Description

Gets the device type from the manufacturer.

Supported on mobile client only.

Syntax

device.of_getdevicetype (ref string *as_type*)

Argument Type	Argument	Description
eon_mobile_deviceex	*device*	The name of the eon_mobile_deviceex object.
string	*as_type*	Returns the device type; e.g., "iPod touch", "iPhone", "iPad", "GT-P5210", "Lenovo A3000-H" etc. The value of device type is set by the manufacturer. Returns empty string if it is called in PowerBuilder or PowerServer Web or if there is any error.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getdpi

Description

Gets the device DPI value.

Supported on mobile client only.

Syntax

device.of_getdpi ()

Argument Type	Argument	Description
eon_mobile_deviceex	*device*	The name of the eon_mobile_deviceex object.

Return value

Integer.

>0 - Return the device DPI value.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getppi

Description

Gets the device PPI value.

Supported on mobile client only.

Syntax

```
device.of_getppi ( )
```

Argument Type	Argument	Description
eon_mobile_deviceex	*device*	The name of the eon_mobile_deviceex object.

Return value

Integer.

>0 - Return the device PPI value.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_gettotalmemory

Description

Gets the total number (in KB) of the device memory.

Supported on mobile client only.

Syntax

```
device.of_gettotalmemory ( )
```

Argument Type	Argument	Description
eon_mobile_deviceex	*device*	The name of the eon_mobile_deviceex object.

Return value

Long.

>0 - Return the total number of the device memory.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getfreememory

Description

Gets the total number (in KB) of the available memory of the device.

Supported on mobile client only.

Syntax

```
device.of_getfreememory ( )
```

Argument Type	Argument	Description
eon_mobile_deviceex	*device*	The name of the eon_mobile_deviceex object.

Return value

Long.

>0 - Return the total number of the device free memory.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getname

Description

Gets the device's machine name.

Supported on mobile client only.

Syntax

device.of_getname (ref string *as_name*)

Argument Type	Argument	Description
eon_mobile_deviceex	*device*	The name of the eon_mobile_deviceex object.
string	*as_name*	Returns the device's machine name. Returns empty string if it is called in PowerBuilder or PowerServer Web, or if there is any error.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getorientation

Description

Detects if the device screen orientation is Landscape or Portrait.

Supported on mobile client only.

Syntax

device.of_getorientation ( )

Argument Type	Argument	Description
eon_mobile_deviceex	*device*	The name of the eon_mobile_deviceex object.

Return value

Integer.

0 - Unknown.

1 - Portrait, Home button at the bottom.

2 - Portrait, Home button on the top.

3 - Landscape, Home button to the right.

4 - Landscape, Home button to the left.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getplatform

Description

Gets the device's OS name.

Supported on mobile client only.

Syntax

device.of_getplatform (ref string *as_platform*)

Argument Type	Argument	Description
eon_mobile_deviceex	*device*	The name of the eon_mobile_deviceex object.
string	*as_platform*	Returns the device's OS name; e.g., returns "iPhone OS" if the device is iPhone, iPad, or iPod touch, and returns "Android" if the device is Android tablets or smartphones. The value of OS name is obtained from the mobile device. Returns empty string if it is called in PowerBuilder or PowerServer Web, or if there is any error.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getresolution

Description

Gets the device's resolution.

Supported on mobile client only.

Syntax

device.of_getresolution (ref integer *ai_height*, ref integer *ai_width*)

Argument Type	Argument	Description
eon_mobile_deviceex	*device*	The name of the eon_mobile_deviceex object.
integer	*ai_height*	Returns the device's screen height (in pixels).
integer	*ai_width*	Returns the device's screen width (in pixels).

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getosversion

Description

Gets the OS version.

Supported on mobile client only.

Syntax

device.of_getosversion (ref string *as_version*)

Argument Type	Argument	Description
eon_mobile_deviceex	*device*	The name of the eon_mobile_deviceex object.
string	*as_version*	Returns the device's OS version. For example, returns "6.0" if the iOS version is 6.0; returns "6.0.1" if the iOS version is 6.0.1. The value of OS version is obtained from the mobile device. Returns empty string if it is called in PowerBuilder or PowerServer Web, or if there is any error.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getstatusbarvisible

Description

Detects if the system status bar is visible.

Supported on mobile client only.

Syntax

device.of_getstatusbarvisible()

Argument Type	Argument	Description
eon_mobile_deviceex	*device*	The name of the eon_mobile_deviceex object.

Return value

Integer.

1 - The system status bar is visible.

0 - The system status bar is invisible.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_setstatusbarvisible

Description

Sets whether the iOS system status bar is visible. This API takes no effect for the Android device and returns -1.

Supported on the iOS mobile client only.

Syntax

device.of_setstatusbarvisible (value integer *ai_mode*)

Argument Type	Argument	Description
eon_mobile_deviceex	*device*	The name of the eon_mobile_deviceex object.
integer	*ai_mode*	1 - Sets the iOS system status bar to be visible. 0 - Sets the iOS system status bar to be invisible.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

2.4.7 GPS

Opens the GPS functionality and gets the current position. You would need to make sure GPS is enabled on the mobile device, and there is good network connection.

Usage example

- Express & delivery services:

  Track the detailed parcel delivery progress: the time and place of departure, the real-time positioning of the delivery men, passage and arrival, etc.

- Geomarketing:

  The online advertising can use this type of marketing, relying on IP addresses to locate the user logged into the Website, and then customizing advertising based on this information.

- Public transportation:

  If a user doesn't know the best way to get to somewhere, the user can use the GPS API to find out nearby bus stops and metro locations.

- Tourism:

  By combining with Map API, travel agencies can provide better vacation tips for the current location and season.

2.4.7.1 eon_mobile_geolocationex object

2.4.7.1.1 Properties

integer ii_errorcode

Error code which is returned when the oe_error event is triggered.

10 - kCLErrorLocationUnknown. Location is currently unknown, but CL will keep trying.

11 - kCLErrorDenied. CL access has been denied (eg, user declined location use).

12 - kCLErrorNetwork. General network-related error.

13 - kCLErrorHeadingFailure. Heading could not be determined.

14 - kCLErrorRegionMonitoringDenied. Location region monitoring has been denied by the user.

15 - kCLErrorRegionMonitoringFailure. A registered region cannot be monitored.

16 - kCLErrorRegionMonitoringSetupDelayed. CL could not immediately initialize region monitoring.

17 - kCLErrorRegionMonitoringResponseDelayed. While events for this fence will be delivered, delivery will not occur immediately.

18 - kCLErrorGeocodeFoundNoResult. A geocode request yielded no result.

19 - kCLErrorGeocodeFoundPartialResult. A geocode request yielded a partial result.

20 - kCLErrorGeocodeCanceled. A geocode request was cancelled.

21 - kCLErrorDeferredFailed. Deferred mode failed.

22 - kCLErrorDeferredNotUpdatingLocation. Deferred mode failed because location updates disabled or paused.

23 - kCLErrorDeferredAccuracyTooLow. Deferred mode not supported for the requested accuracy.

24 - kCLErrorDeferredDistanceFiltered. Deferred mode does not support distance filters.

25 - kCLErrorDeferredCanceled. Deferred mode request canceled a previous request.

powerobject ipo_bindevent

The object to bind with the oe_locationchanged event and oe_error event.

string is_bindwitherrorevent

The name of the event bound with the oe_error event of the powerobject ipo_bindevent.

string is_bindwithlocationchangedevent

The name of the event bound with the oe_locationchanged event of the powerobject ipo_bindevent.

string is_errortext

Error message which is returned when the oe_error event is triggered.

2.4.7.1.2 Events

oe_error

Description

It will be triggered automatically if there is an error when using the GPS functionality.

Supported on mobile client only.

Note: This event will update the value of is_errortext and ii_errorcode properties.

Syntax

None.

Return value

None.

oe_locationchanged

Description

It will be triggered automatically when the position (longitude and/or latitude) changes after getting the GPS information.

Supported on mobile client only.

Note: The user can obtain the new position information by calling the of_getcurrentposition function. WPARAM and LPARAM are both empty.

When the messages for this event in the message queue reaches 500, the new messages for this event will not be stored, they will be directly discarded.

Syntax

oe_locationchanged (decimal *dec_longitude*, decimal *dec_latitude*, decimal *dec_altitude*, decimal *dec_accuracy*, decimal *dec_altitudeaccuracy*, decimal *dec_heading*, decimal *dec_speed*, datetime *dt_timestamp*)

Parameter Name	Parameter Type	Description
dec_longitude	Decimal	The longitude value of the current position.
dec_latitude	Decimal	The latitude value of the current position.
dec_altitude	Decimal	The altitude value of the current position.
dec_accuracy	Decimal	The latitude and longitude positioning accuracy.
dec_altitudeaccuracy	Decimal	The altitude positioning accuracy.
dec_heading	Decimal	The degrees clockwise from true north (0 to 359.99 degrees).
dec_speed	Decimal	The displacement velocity (m/sec).
dt_timestamp	DateTime	The time stamp to get the current position information.

Return value

None.

2.4.7.1.3 Functions

of_close

Description

Closes the GPS functionality.

Supported on mobile client only.

Syntax

gps.of_close ( )

Argument Type	Argument	Description
eon_mobile_geolocationex	*gps*	The name of the eon_mobile_geolocationex object.

Return value

Integer.

1 - Success.

-1 - If it is called in PowerBuilder or PowerServer Web, or there is an error.

of_getcurrentposition

Description

Gets the current position information. Be sure to call this function separately from the of_open function, otherwise, this function will not work as expected. The best practice is to call this function in a user event which is bound with the oe_locationchanged event, because the oe_locationchanged event will be automatically triggered once the of_open function is finished. The recommended steps are:

1. Call the of_register function to bind the user event with the oe_locationchanged event.
2. Call the of_open function.
3. In the user event that is bound in Step 1, call the of_getcurrentposition function.

Supported on mobile client only.

Syntax

gps.of_getcurrentposition (ref eon_mobile_str_coordinates *astr_coordinates*)

Argument Type	Argument	Description
eon_mobile_geolocationex	*gps*	The name of the eon_mobile_geolocationex object.
eon_mobile_str_coordinates	*astr_coordinates*	The returned detailed position information. See variable list of eon_mobile_str_coordinates.

Return value

Integer.

1 - Success.

-1 - If it is called in PowerBuilder or PowerServer Web, or there is an error.

Code example

The following code example shows you how to call of_getcurrentposition and of_open functions separately (in different events), in order for of_getcurrentposition to work normally.

```
//instance variables of w_1 window
   eon_mobile_geolocationex  ignv_aws
```

```
//in the constructor event of w_1 window
```

```
……
if appeongetclienttype()="MOBILE" then
   ignv_aws = CREATE eon_mobile_geolocationex
      If ignv_aws.of_isenabled() = 1 Then
         ignv_aws.of_open (0, 1)
      Else
         destroy ignv_aws
         messagebox('', 'GPS Disabled')
      End If
end if
……
```

```
// in clicked event of cb_1 of w_1 window
……
if isvalid( ignv_aws) then
   ignv_aws.of_getcurrentposition (astr_coordinates)
   ignv_aws.of_close()
   messagebox('Long',astr_coordinates.dec_longitude)
   messagebox('Long',astr_coordinates.dec_latitude)
end if
……
```

The following code example shows you how to call of_open and of_getcurrentposition in the same event. But this code example only works in the iOS application, not in the Android application.

```
if appeongetclienttype()="MOBILE" then
   eon_mobile_geolocationex  lgnv_aws
       lgnv_aws = CREATE eon_mobile_geolocationex
       If lgnv_aws.of_isenabled() = 1 Then
              lgnv_aws.of_open (0, 1)
           sleep(1)
              lgnv_aws.of_getcurrentposition (astr_coordinates)
              lgnv_aws.of_close()
              messagebox('Long',astr_coordinates.dec_longitude)
              messagebox('Long',astr_coordinates.dec_latitude)
        Else
              messagebox('', 'GPS Disabled')
        End If
    destroy lgnv_aws
end if
```

of_isenabled

Description

Detects if the GPS service can be used.

Supported on mobile client only.

Syntax

gps.of_isenabled()

Argument Type	Argument	Description
eon_mobile_geolocationex	*gps*	The name of the eon_mobile_geolocationex object.

None.

Return value

Integer

1 - Enabled.

0 - Disabled.

-1 - If it is called in PowerBuilder or PowerServer Web, or there is an error.

of_open

Description

Opens the GPS functionality. The execution time of this function relies on the network connection between the mobile device and the GPS satellites. Only after this function is finished, can of_getcurrentposition work normally.

Supported on mobile client only.

Syntax

gps.of_open (value integer *ai_locationaccuracy*, value integer *ai_distancefilter*)

Argument Type	Argument	Description
eon_mobile_geolocationex	*gps*	The name of the eon_mobile_geolocationex object.
integer	*ai_locationaccuracy*	Location accuracy update, recommended to use 0 or 1 for the automatic selection. 0 - Use the highest-level of accuracy. 1 - Use the highest possible accuracy and combine it with additional sensor data. This level of accuracy is intended for use in navigation applications that require precise position information at all times and are intended to be used only while the device is plugged in. >1 - User-defined accuracy (in meters).
integer	*ai_distancefilter*	Location filter, used to control the location update message frequency (in meters). 0 - Notifies by every update.

Argument Type	Argument	Description
		>0 - Updates only when the location change exceeds this value.

Return value

Integer.

1 - Success.

-1 - If it is called in PowerBuilder or PowerServer Web, or there is an error.

of_register

Description

Binds user-defined events with the oe_error event and the oe_locationchanged event.

After the oe_error event and the oe_locationchanged event are triggered, the bound events will be triggered automatically. The bound events can be utilized to extend the events of the GPS object (which is an NVO object), as NVO object cannot be extended in the PB IDE. If you do not want to extend the events, then you do not need to call the of_register function, as it will be called internally and automatically.

Supported on mobile client only.

Syntax

gps.of_register (value powerobject *apb_bind*, value string *as_changedevent*, value string *as_errorevent*)

Argument Type	Argument	Description
eon_mobile_geolocationex	*gps*	The name of the eon_mobile_geolocationex object.
powerobject	*apb_bind*	The object to be bound with the oe_error event and oe_locationchanged event.
string	*as_changedevent*	The event to be bound with the oe_locationchanged event. If this parameter is set to null, oe_locationchanged event of the bound object will be triggered; in such case, make sure the bound object has the oe_locationchanged event and the same parameter amount and type.
string	*as_errorevent*	The event to be bound with the oe_error event.

Return value

Integer.

1 - Success.

-1 - If it is called in PowerBuilder or PowerServer Web, or there is an error.

of_triggerevent

Description

This is an internal function.

2.4.7.1.4 Structures

eon_mobile_str_coordinates

Description

Struct.

The detailed position information.

Property

Type	Variable Name	Description
decimal{6}	`dec_longitude`	The longitude value of the current position.
decimal{6}	`dec_latitude`	The latitude value of the current position.
decimal{2}	`dec_altitude`	The altitude value of the current position.
decimal{2}	`dec_accuracy`	The latitude and longitude positioning accuracy.
decimal{2}	`dec_altitudeaccuracy`	The altitude positioning accuracy.
decimal{2}	`dec_heading`	The degrees clockwise from true north (0 to 359.99 degrees).
decimal{2}	`dec_speed`	The displacement velocity (m/sec).
datetime	`dt_timestamp`	The time stamp to get the current position information.

2.4.7.1.5 Code Example

- To get the GPS information of the photo you take:

```
eon_mobile_cameraex inv_camera
eon_mobile_geolocationex inv_gps
```

```
inv_gps = create eon_mobile_geolocationex
inv_camera = create eon_mobile_cameraex

// opens the geolocation function
inv_gps.of_open (3, 0)

// take a photo
li_return = inv_camera.of_takefile (1, false, ls_filepath)

if li_return = 1 then
…
    // get current location info for the photo
inv_gps.of_getcurrentposition (istr_coordinates)
…
end if
```

- To mark the movement on the map:

  1. Adds an eon_mobile_mapex object -- uo_map -- to the window.
  2. Declares an instance variable.

     ```
     eon_mobile_geolocationex inv_gps
     ```

  3. Registers the GPS service with the Open event of the window.

     ```
     inv_gps = create eon_mobile_geolocationex
     eon_mobile_str_mapoption lstr_mapoption

     // Sets the default argument of the map
     lstr_mapoption.b_allowmove = true
     lstr_mapoption.b_allowzoom = true
     lstr_mapoption.b_locatetocurrentlocation = true
     lstr_mapoption.i_mapaccuracy = 5

     // Opens the map
     uo_map.of_open (lstr_mapoption)

     // The ue_gps event will be triggered when the positioning accuracy is 3
     // meters and the location update range is beyond 100 meters.
     inv_gps.of_open (3, 100)

     // Bind the "ue_gps" event and "ue_gpserror" event of the Window to
     // the "oe_locationchanged" event and "oe_error" event of Geolocation object.
     inv_gps.of_register (this, "ue_gps", "ue_gpserror")
     ```

  4. Records the movement and adds it as an annotation to the map.

     ```
     // ue_gps event

     eon_mobile_str_coordinates lstr_coordinates
     eon_mobile_str_annotation lstr_annotation_appeon

     // Get the gps data of the current position
     inv_gps.of_getcurrentposition (lstr_coordinates)

     // Displays the latitude and longitude of the position in the static text
     st_latitude.text = string (lstr_coordinates.dec_latitude)
     st_longitude.text = string (lstr_coordinates.dec_longitude)

     // Records the gps information of the current position to an annotation
     lstr_annotation_appeon.dec_latitude= lstr_coordinates.dec_latitude
     lstr_annotation_appeon.dec_longitude = lstr_coordinates.dec_longitude
     ```

```
lstr_annotation_appeon.i_pincolor = 1
lstr_annotation_appeon.s_title= string (lstr_coordinates.dt_timestamp,
 "hh:mm:ss")

// Adds this annotation to the map
uo_map.of_addannotation (lstr_annotation_appeon, true)
```

5. Captures the GPS error.

```
// ue_gpserror event
// Displays the error number and the error message
messagebox ("Error "+string(inv_gps.ii_errorcode), inv_gps.is_errortext)
```

2.4.8 Image

Sets the quality or the gray level of an image, or generates a thumbnail of the image with specified options.

2.4.8.1 eon_mobile_imageex object

2.4.8.1.1 Functions

of_createimagewithoptions

Description

Generates an image according to the specified options.

Supported on mobile client only.

Syntax

image.`of_createimagewithoptions` (blob ***`ablb_sourcefile`***, string ***`as_targetfile`***, eon_mobile_str_imageoption ***`astr_imageoption`***, ref eon_mobile_str_imageinfo ***`astr_imageinfo`***)

image.`of_createimagewithoptions` (string ***`as_sourcefile`***, string ***`as_targetfile`***, eon_mobile_str_imageoption ***`astr_imageoption`***, ref eon_mobile_str_imageinfo ***`astr_imageinfo`***)

Argument Type	Argument	Description
eon_mobile_imageex	***`image`***	The name of the eon_mobile_imageex object.
blob	***`ablb_sourcefile`***	The source file of the image (Blob data).
string	***`as_sourcefile`***	The location and file name of the source image.
string	***`as_targetfile`***	The location and file name of the generated image. When file name is empty, no image will be generated, only the blob data will be generated, and if `b_returnblob` of **eon_mobile_str_imageoption** is false, the function will return an argument error.
eon_mobile_str_imageoption	***`astr_imageoption`***	The arguments for generating the target image. See variable list of eon_mobile_str_imageoption.
eon_mobile_str_imageinfo	***`astr_imageinfo`***	The information of the generated image.

Argument Type	Argument	Description
		See variable list of eon_mobile_str_imageinfo.

Return value

Integer

1 - Successful.

0 - Unknown error.

-1 - Invalid file location.

-2 - Argument error.

-3 - Failed to load the source image file.

-4 - Failed to save the generated image file.

-5 - Failed to compress the image to the specified size.

-6 - Insufficient memory.

-7 - File already exists.

2.4.8.1.2 Structures

eon_mobile_str_imageoption

Description

The options for generating the target image.

Property

Type	Variable Name	Description
long	`l_width`	The width of the target image. If 0, then the original width will be used to generate the image.
long	`l_height`	The height of the target image. If 0, then the original height will be used to generate the image.
integer	`i_quality`	The quality of the target image: [1, 10]. 10 indicates the original quality. If it is a different number other than 10, it indicates the quality of the target image and only JPG format will be supported and i_imagetype will be ignored.
double	`dbl_filesize`	When `dbl_filesize` (same as Blob, unit is BYTE) is greater than zero, `i_quality` will be ignored, and the image quality will be automatically adjusted according to the value of `i_loopstep`, and therefore only JPG image type will be supported.
integer	`i_loopstep`	The percentage of the quality loss during compression: [1-9]. The number indicates the percentage level (1 is 10%, 9 is 90% etc.) that the quality will be reduced by.

Type	Variable Name	Description
boolean	b_aspectfit	Keep the aspect ratio during compression. When it is true, keep the aspect ratio. If the width aspect ratio (the target image width divided by the source image width) is less than the height aspect ratio (the target image height divided by the source image height), then the width aspect ratio will be used as the compression ratio, otherwise, the height aspect ratio will be used as the compression ratio.
boolean	b_returnblob	Whether to return the indicator of the blob data.
boolean	b_grayimage	Whether to gray out the target image.
boolean	b_existoverwrite	Whether to overwrite the existing file.
integer	i_imagetype	The supported image file type: 0 - JPG, & 1 - PNG
boolean	b_scalefit	When true, the image will be generated at the size of the content area (the image will be scaled to fill up the content area, possibly anamorphically); when false, the image will be generated at the size specified by l_width and l_height and will not be scaled.

eon_mobile_str_imageinfo

Description

The information of the generated image.

Property

Type	Variable Name	Description
long	l_width	The width of the generated image.
long	l_height	The height of the generated image.
integer	i_quality	The quality of the generated image: [1, 10]. 10 indicates the original quality.
double	dbl_filesize	The size of the generated image.
blob	ablb_filedata	The data of the generated image, only when b_returnblob of **eon_mobile_str_imageoption** is true, there is value of ablb_filedata.

2.4.9 Map

Accesses the information about the longitude, latitude, and address as well as adds/removes annotations.

Note: In order to load the map on the Android device, you would need to make sure the Google Play service is installed beforehand.

Usage example

- City planning:

  Use the Street View maps to implement a better city planning and design.

- Fire department:

  The fire control center can find out the specific position of the man who made the emergency call according to the system, and then quickly plan their arrangement to send the nearest firefighters.

- Mountaineering:

  For mountain-climbing lovers, they can identify a location according to the known longitude and latitude and vice versa.

- Tourism:

  By combining with GPS API, travel agencies can provide better vacation tips for the current location and season.

- Turn-by-turn route navigation:

  Help the user navigate along a route by providing detailed turn-by-turn directions. Map API does this by registering with the GPS API to receive repeated location updates of the user's position.

2.4.9.1 eon_mobile_mapex object

2.4.9.1.1 Properties

eon_mobile_str_annotation istr_clickedannotation

The annotation (added by of_addannotation) that the user clicks on.

integer ii_errorcode

Error code which is returned when the oe_error event is triggered.

1 - MKErrorUnknown

2 - MKErrorServerFailure. The map service is turned off.

3 - MKErrorLoadingThrottled

4 - MKErrorPlacemarkNotFound. Address is not found.

-1 - NSURLErrorUnknown

-999 - NSURLErrorCancelled

-1001 - NSURLErrorTimedOut

-1002 - NSURLErrorCannotFindHost

-1004 - NSURLErrorCannotConnectToHost

-1005 - NSURLErrorNetworkConnectionLost

-1009 - NSURLErrorNotConnectedToInternet

string is_errortext

Error message which will be returned when the oe_error event is triggered.

2.4.9.1.2 Events

oe_annotationclicked

Description

This event will be automatically triggered when an annotation (added by of_addannotation) is clicked.

Supported on mobile client only.

Note: This event cannot be triggered via the code, and can only be triggered by clicking the annotation on the map.

Syntax

None.

Return value

None.

The information of the clicked annotation is stored to the istr_clickedannotation instance variable.

oe_loadstart

Description

This event will be automatically triggered when starting the loading of the map. LPARAM and WPARAM are empty.

Supported on mobile client only.

Note: This event is automatically triggered after calling the of_open function.

Syntax

None.

Return value

None.

oe_loadend

Description

This event is automatically triggered upon the successful completion of loading the map. LPARAM and WPARAM are empty.

Supported on mobile client only.

Note: This event is automatically triggered after calling the of_open function.

Syntax

None.

Return value

None.

oe_error

Description

This event is automatically triggered upon failure in loading the map. WPARAM is the error number.

Supported on mobile client only.

Note: This event is automatically triggered after calling the of_open function.

This event will update the value of is_errortext and ii_errorcode properties.

Syntax

None.

Return value

None.

oe_changed

Description

This event is automatically triggered when the map display area has changed.

Note: This event is automatically triggered after calling the of_open function.

Syntax

None.

Return value

None.

2.4.9.1.3 Functions

of_addannotation

Description

Adds a map annotation.

Supported on mobile client only.

Syntax

map.of_addannotation (ref eon_mobile_str_annotation *astr_annotation*, value boolean *ab_move*)

Argument Type	Argument	Description
eon_mobile_mapex	*map*	The name of the eon_mobile_mapex object.
eon_mobile_str_annotation	*astr_annotation*	The annotation to be added. (For the relevant information, refer to the property of the

Argument Type	Argument	Description
		eon_mobile_str_annotation structure.) When the annotation is added successfully, this function will generate and store the unique identifier of the annotation to the l_id property in the structure. And the l_id property will be used when this annotation is removed or selected.
boolean	*ab_move*	True: To automatically move the map so as to make the new annotation visible. False: Not to move the map.

Return value

Integer.

1 - Success.

-1 - Failure or it is called in PowerBuilder or PowerServer Web, or there is an error.

Code example

```
eon_mobile_str_annotation istr_annotation_appeon
istr_annotation_appeon.dec_latitude= 22.54993
istr_annotation_appeon.dec_longitude = 113.94977
istr_annotation_appeon.i_pincolor = 1
istr_annotation_appeon.s_pinimage="apb.png"
istr_annotation_appeon.s_subtitle="PowerServer Mobile"
istr_annotation_appeon.s_title="Appeon"
uo_map.of_addannotation(istr_annotation_appeon, true)
```

of_addresstocoordinate

Description

Obtains the longitude and latitude according to the address. **Note**: before this function is called, be sure to add an eon_mobile_mapex control and call the of_open function of this control to open the map successfully.

Supported on mobile client only.

Syntax

map.of_addresstocoordinate (value string *as_address*, ref decimal *dec_longitude*, ref decimal *adec_latitude*)

Argument Type	Argument	Description
eon_mobile_mapex	*map*	The name of the eon_mobile_mapex object.

Argument Type	Argument	Description
string	*as_address*	Address, a human-readable string to specify locations.
decimal	*dec_longitude*	The returned longitude, which is defined using numerals within a comma-separated text string that have a precision to 6 decimal places.
decimal	*adec_latitude*	The returned latitude, which is defined using numerals within a comma-separated text string that have a precision to 6 decimal places.

Return value

Integer

1 - Success.

-1 - Failure or it is called in PowerBuilder or PowerServer Web, or there is an error.

of_close

Description

Closes the default map application.

Supported on mobile client only.

Syntax

map.of_close ()

Argument Type	Argument	Description
eon_mobile_mapex	*map*	The name of the eon_mobile_mapex object.

Return value

Integer.

1 - Success.

-1 - Failure or it is called in PowerBuilder or PowerServer Web, or there is an error.

of_coordinatetoaddress

Description

Converts the longitude and latitude into the corresponding address. **Note**: before this function is called, be sure to add an eon_mobile_mapex control and call the of_open function of this control to open the map successfully.

Supported on mobile client only.

Syntax

map.of_coordinatetoaddress (value decimal dec_longitude, value decimal *adec_latitude*, ref string *as_address*)

Argument Type	Argument	Description
eon_mobile_mapex	*map*	The name of the eon_mobile_mapex object.
decimal	*dec_longitude*	Longitude, which is defined using numerals within a comma-separated text string that have a precision to 6 decimal places.
decimal	*adec_latitude*	Latitude, which is defined using numerals within a comma-separated text string that have a precision to 6 decimal places.
string	*as_address*	The returned address, which is converted from longitude and latitude values.

Return value

Integer.

1 - Success.

-1 - Failure or it is called in PowerBuilder or PowerServer Web, or there is an error.

of_open

Description

Opens the default map application at the specified position.

Supported on mobile client only.

Syntax

map.of_open (value eon_mobile_str_mapoption *astr_mapoption*)

map.of_open (value eon_mobile_str_mapoption *astr_mapoption*, ref eon_mobile_str_annotation *astr_clickedannotation*)

Argument Type	Argument	Description
eon_mobile_mapex	*map*	The name of the eon_mobile_mapex object.
eon_mobile_str_mapoption	*astr_mapoption*	Relevant information about the map. (For the relevant information, refer to the property of the

Argument Type	Argument	Description
		eon_mobile_str_mapoption structure.)
eon_mobile_str_annotation	*astr_clickedannotation*	The annotation that the user clicks on. (For the relevant information, refer to the property of the eon_mobile_str_annotation structure.)

Return value

Integer.

1 - Success.

-1 - Failure or it is called in PowerBuilder or PowerServer Web, or there is an error.

Code example

```
eon_mobile_str_mapoption lstr_mapoption
lstr_mapoption.b_allowmove = true
lstr_mapoption.b_allowzoom = true
lstr_mapoption.b_locatetocurrentlocation = true
lstr_mapoption.i_mapaccuracy = 5
uo_map.of_open(lstr_mapoption)
```

of_refresh

Description

Refreshes the map according to the specified information.

Supported on mobile client only.

Syntax

map.of_refresh (value eon_mobile_str_mapoption *astr_mapoption*)

Argument Type	Argument	Description
eon_mobile_mapex	*map*	The name of the eon_mobile_mapex object.
eon_mobile_str_mapoption	*astr_mapoption*	Relevant information about the map. (For the relevant information, refer to the property of the eon_mobile_str_mapoption structure.)

Return value

Integer.

1 - Success.

-1 - Failure or it is called in PowerBuilder or PowerServer Web, or there is an error.

of_removeannotation

Description

Removes a map annotation that is added by of_addannotation.

Supported on mobile client only.

Syntax

map.of_removeannotation (value eon_mobile_str_annotation *astr_annotation*)

map.of_removeannotation (value long *al_id*)

Argument Type	Argument	Description
eon_mobile_mapex	*map*	The name of the eon_mobile_mapex object.
eon_mobile_str_annotation	*astr_annotation*	The annotation to be removed. (For the relevant information, refer to the property of the eon_mobile_str_annotation structure.) This function will remove the annotation according to the unique identifier of the annotation stored in the l_id property of the structure.
long	*al_id*	The unique identifier of the annotation to be removed.

Return value

Integer.

1 - Success.

-1 - Failure or it is called in PowerBuilder or PowerServer Web, or there is an error.

of_removeallannotation

Description

Removes all map annotations that are added by of_addannotation.

Supported on mobile client only.

Syntax

map.of_removeallannotation ( )

Argument Type	Argument	Description
eon_mobile_mapex	*map*	The name of the eon_mobile_mapex object.

Return value

Integer.

1 - Success.

-1 - Failure or it is called in PowerBuilder or PowerServer Web, or there is an error.

of_selectannotation

Description

Selects a map annotation that is added by of_addannotation.

Supported on mobile client only.

Syntax

map.of_selectannotation (value eon_mobile_str_annotation *astr_annotation*)

map.of_selectannotation (value long *al_id*)

Argument Type	Argument	Description
eon_mobile_mapex	*map*	The name of the eon_mobile_mapex object.
eon_mobile_str_annotation	*astr_annotation*	The annotation to be selected. (For the relevant information, refer to the property of the eon_mobile_str_annotation structure.) This function will select the annotation according to the unique identifier of the annotation stored in the l_id property of the structure.
long	*al_id*	The unique identifier of the annotation to be selected.

Return value

Integer.

1 - Success.

-1 - Failure or it is called in PowerBuilder or PowerServer Web, or there is an error.

2.4.9.1.4 Structures

eon_mobile_str_mapoption

Description

Struct.

A map annotation. The user can click on the text markup.

Property

Type	Variable Name	Description
boolean	b_locatetocurrent location	Moves to the current location.
decimal{6}	dec_longitude	The longitude of the map central location (only takes effect when the value of b_locatetocurrent location is False).
decimal{6}	dec_latitude	The latitude of the map central location.
boolean	b_allowmove	Allows the user to move or not.
boolean	b_allowzoom	Allows the user to zoom or not.
integer	i_mapaccuracy	The map display accuracy or amplification factor (unit: meter).
integer	i_maptype	The map type (0: Map (Default), 1: Satellite, 2: Mixed).

eon_mobile_str_annotation

Description

Struct.

Relevant information about the initialized map.

Property

Type	Variable Name	Description
decimal{6}	dec_longitude	The longitude of the added or obtained annotation.
decimal{6}	dec_latitude	The latitude of the added or obtained annotation.
string	s_title	The title of the added or obtained annotation.
string	s_subtitle	The subtitle of the added or obtained annotation.
integer	i_pincolor	The pin color. Only takes effect when there is no user-defined pin icon (the

Type	Variable Name	Description
		supported colors are: Red = 0, Green = 1, Purple = 2).
string	s_pinimage	The pin image (If it is NULL, use the default icon).
long	l_id	The unique identifier of the annotation.

2.4.10 Media

Plays a media file using the device's media player.

2.4.10.1 eon_mobile_mediaex object

2.4.10.1.1 Functions

of_play

Description

Uses the default media player application to play the media files.

The supported audio file formats are .aac, .mp3, .aiff, and .wav; the supported video file formats are .m4v, .mp4, and .mov.

Not all the media files with these suffixes can be played; if some media files cannot be played, they can be transcoded via iTunes.

Supported on mobile client only.

Syntax

media.of_play (value string *as_filepath*)

Argument Type	Argument	Description
eon_mobile_mediaex	*media*	The name of the eon_mobile_mediaex object.
string	*as_filepath*	Sets the full path of a media file to be played.

Return value

Integer.

1 - Play the media file successfully.

0 - Cancel playing the media file.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

2.4.11 NFC

Provides access to Near Field Communication (NFC) functionality, allowing the applications to read NDEF message from the Android device.

Note: iOS devices have no NFC features at all, therefore, the NFC APIs are for the Android-powered device only, however, not every Android-powered device provides NFC functionality. For more information, see the Near Field Communication guide.

There are two methods to read/write data from the NFC tag. Pay attention to the functions listed below, because they are effective for one method only. The functions not listed below are effective for both methods, such as of_open, of_close, of_getblocksize, of_getmaxblockcount, of_getmaxsectorcount, of_getsize, of_gettagid.

- Method 1: Read/write data in NDEF (NFC Data Exchange Format)

  The following functions are effective for this method only:

  of_getndefrecordcount

  of_getndefrecord

  of_addrecord

  of_writerecords

- Method 2: Read/write data in bytes from blocks

  The following functions are effective for this method only:

  of_settimeout

  of_gettimeout

  of_setkey

  of_startblockoperate

  of_stopblockoperate

  of_readblock

  of_writeblock

See the flowchart and code examples for method 1 and 2 in Code example.

2.4.11.1 eon_mobile_nfcex object

2.4.11.1.1 Properties

integer ii_errorcode

Error code which is returned when the oe_error event is triggered.

0 - Exception. Exception occurs during operation. The error message will be returned by the Android OS.

1 - ServiceDisable. NFC service is not enabled or the current device does not support NFC.

2 - DomainInvalid. Domain is invalid or contains illegal characters such as colon.

3 - MIMETypeInvalid. MIME type cannot be empty or contain multi-byte characters.

integer ii_nfccardtype

NFC card type.

0 - MifareClassic. Requires the encryption key, or uses the default key.

1 - MifareUltralight. Requires no encryption key.

2 - NfcV. Requires no encryption key.

3 - NDEFOnly. Supports read/write NDEF records but does not support read/write data (in bytes) from the block.

4 - UNKNOWN. Unrecognizable tag types. Cannot read/write from the tag, but can get tag identifier.

powerobject ipo_bindevent

The object to bind with the oe_newtagfound event and oe_error event.

string is_bindwitherrorevent

The name of the event bound with the oe_error event of the powerobject ipo_bindevent.

string is_bindwithnewtagfoundevent

The name of the event bound with the oe_newtagfound event of the powerobject ipo_bindevent.

string is_errortext

Error message which is returned when the oe_error event is triggered.

2.4.11.1.2 Events

oe_newtagfound

Description

Automatically occurs when the NFC tag is detected by the reader. **Note**: This event is triggered internally and automatically.

Syntax

`nfc.oe_newtagfound ( integer ai_type )`

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.
integer	ai_type	0 - MifareClassic. Requires the encryption key, or uses the default key. 1 - MifareUltralight. Requires no encryption key. 2 - NfcV. Requires no encryption key. 3 - NDEFOnly. Supports read/write NDEF records but does not support read/write data (in bytes) from the block. 4 - UNKNOWN. Unrecognizable tag types. Cannot read/write from the tag, but can get tag identifier.

Return value

None

oe_error

Description

It will be triggered automatically if there is an error when using the NFC functionality.

Supported on mobile client only.

Note: This event will update the value of is_errortext and ii_errorcode properties.

Syntax

None.

Return value

None.

2.4.11.1.3 Functions

of_addrecord

Description

Adds the NDEF record.

Supported on mobile client only.

Syntax

nfc.of_addrecord (eon_mobile_str_nfcrecord *astr_nfcrecord*)

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.
eon_mobile_str_nfcrecord	*astr_nfcrecord*	The record to be added. See variable list of eon_mobile_str_nfcrecord.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_close

Description

Closes the NFC communication.

Supported on mobile client only.

Syntax

```
nfc.of_close ()
```

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getblocksize

Description

Gets the size (in bytes) of the block.

Supported on mobile client only.

Syntax

```
nfc.of_getblocksize ()
```

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.

Return value

Integer.

>0 - Returns the size (in bytes) of the block.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getmaxblockcount

Description

Gets the total number of blocks.

Supported on mobile client only.

Syntax

```
nfc.of_getmaxblockcount ()
```

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.

Return value

Integer.

>0 - Returns the total number of blocks.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getmaxsectorcount

Description

Gets the total number of sectors.

Supported on mobile client only.

Syntax

`nfc.of_getmaxsectorcount ()`

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.

Return value

Integer.

>0 - Returns the total number of sectors.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getndefrecord

Description

Gets the NDEF record. The records cannot be read right after they are written to the tag; the tag will need to be detected by the reader again, and then the records can be read by using this function.

Supported on mobile client only.

Syntax

`nfc.of_getndefrecord ( integer ai_index, ref eon_mobile_str_nfcrecord astr_nfcrecord )`

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.
integer	*ai_index*	The record index which is obtained through of_getndefrecordcount. Record indexes are sequential numbers starting from 1.

Argument Type	Argument	Description
eon_mobile_str_nfcrecord	*astr_nfcrecord*	The record to be read. See variable list of eon_mobile_str_nfcrecord.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getndefrecordcount

Description

Gets the total number of the NDEF records.

Supported on mobile client only.

Syntax

```
nfc.of_getndefrecordcount ()
```

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.

Return value

Integer.

>0 - Returns the total number of the NDEF records.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getsize

Description

Gets the total capacity (in bytes) of the tag.

Supported on mobile client only.

Syntax

```
nfc.of_getsize ()
```

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.

Return value

Integer.

>0 - Returns the total capacity (in bytes) of the tag.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_gettagid

Description

Gets the unique identifier of the tag.

Supported on mobile client only.

Syntax

nfc.of_gettagid (ref string *as_tagid*)

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.
string	*as_tagid*	Returns the unique identifier of the tag; or returns empty string if it is called in PowerBuilder or PowerServer Web or if there is any error.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_gettimeout

Description

Gets the timeout value for reading/writing data.

Supported on mobile client only.

Syntax

nfc.of_gettimeout ()

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.

Return value

Integer.

>0 - Returns the timeout value for reading/writing data.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_open

Description

Starts the NFC communication. Make sure this function is called before the tag is detected, otherwise, the other functions (such as the read/write functions) will not take effect. If the tag is detected before this function is called, the tag will need to be detected again after this function is called.

Supported on mobile client only.

Syntax

```
nfc.of_open ()
```

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_readblock

Description

Reads data (in bytes) from the block. Make sure of_startblockoperate is called before the tag is detected and this function is called.

The records cannot be read right after they are written to the tag; the tag will need to be detected by the reader again, and then the records can be read by using the of_readblock function.

Supported on mobile client only.

Syntax

```
nfc.of_readblock ( integer ai_block, ref blob abl_data )
```

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.
integer	*ai_block*	The block index which is obtained through of_getMaxBlockCount. Block indexes are sequential numbers starting from 1.
blob	*abl_data*	Data that are read from the block.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_register

Description

Binds user-defined events with the oe_newtagfound event and the oe_error event.

After oe_newtagfound and oe_error are triggered, the bound events will be triggered automatically. The bound events can be utilized to extend the events of the NFC object, as the NFC object is an NVO object which cannot be extended in the PB IDE. If you do not want to extend the events, then you do not need to call the of_register function, as it will be called internally and automatically.

Supported on mobile client only.

Syntax 1

nfc.of_register (powerobject *apb_bind*, string *as_newtagfound*)

Syntax 2

nfc.of_register (powerobject *apb_bind*, string *as_newtagfound*, string *as_error*)

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.
powerobject	*apb_bind*	The object to be bound with the oe_newtagfound event and the oe_error event.
string	*as_newtagfound*	The event to be bound with the oe_newtagfound event.
string	*as_error*	The event to be bound with the oe_error event.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_setkey

Description

Sets the read/write encryption key.

Supported on mobile client only.

Syntax

nfc.of_setkey (string *as_key*)

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.
string	*as_key*	Encryption key for reading/writing data. Or use the system default key. MIFARE Classic tags require the key. Tag types can be obtained through oe_newtagfound event.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_settimeout

Description

Sets the timeout value for reading/writing data.

Supported on mobile client only.

Syntax

nfc.of_settimeout (integer *ai_timeout*)

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.
integer	*ai_timeout*	Timeout value for reading/writing data. Normally, this parameter needs not to be set. Only if the read/write operation would take a long time, then this parameter should be set, otherwise the operation will time out and disconnect automatically.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_startblockoperate

Description

Starts reading/writing data from the block. This function must be called before of_writeblock or of_readblock is called.

Supported on mobile client only.

Syntax

```
nfc.of_startblockoperate ()
```

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_stopblockoperate

Description

Stops reading/writing data from the block.

Supported on mobile client only.

Syntax

```
nfc.of_stopblockoperate ()
```

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_triggerevent

Description

Triggers the object events bound with the oe_newtagfound event.

Supported on mobile client only.

Syntax

nfc.of_triggerevent (string *as_event*)

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.
string	*as_event*	The event name bound with oe_newtagfound event.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_writeblock

Description

Writes data (in bytes) to the block. Before this function is called, make sure of_startblockoperate is called first, otherwise this function will not take effect.

After the records are successfully written to the tag, they cannot be read immediately, the tag will need to be detected by the reader again, and then the records can be read successfully by using the of_readblock function.

Supported on mobile client only.

Syntax

nfc.of_writeblock (integer *ai_block*, blob *abl_data*)

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.
integer	*ai_block*	The block index which is obtained through of_getMaxBlockCount. Block indexes are sequential numbers starting from 1.
blob	*abl_data*	Data to be written to the block. Block size should be obtained through of_getBlockSize first. Block size can be 4 bytes or 16 bytes. Data exceeding the block size will be automatically truncated.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_writerecords

Description

Writes the NDEF record. of_addrecord should be called first to add one or more records, then of_writerecords should be called to write all records at one time. After the records are successfully written to the tag, they cannot be read immediately, the tag will need to be detected by the reader again, and then the records can be read successfully by using the of_getndefrecord function.

Supported on mobile client only.

Syntax

```
nfc.of_writerecords ()
```

Argument Type	Argument	Description
eon_mobile_nfcex	*nfc*	The name of the eon_mobile_nfcex object.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

2.4.11.1.4 Structures

eon_mobile_str_nfcrecord

Description

Structure.

The record to be added.

Property

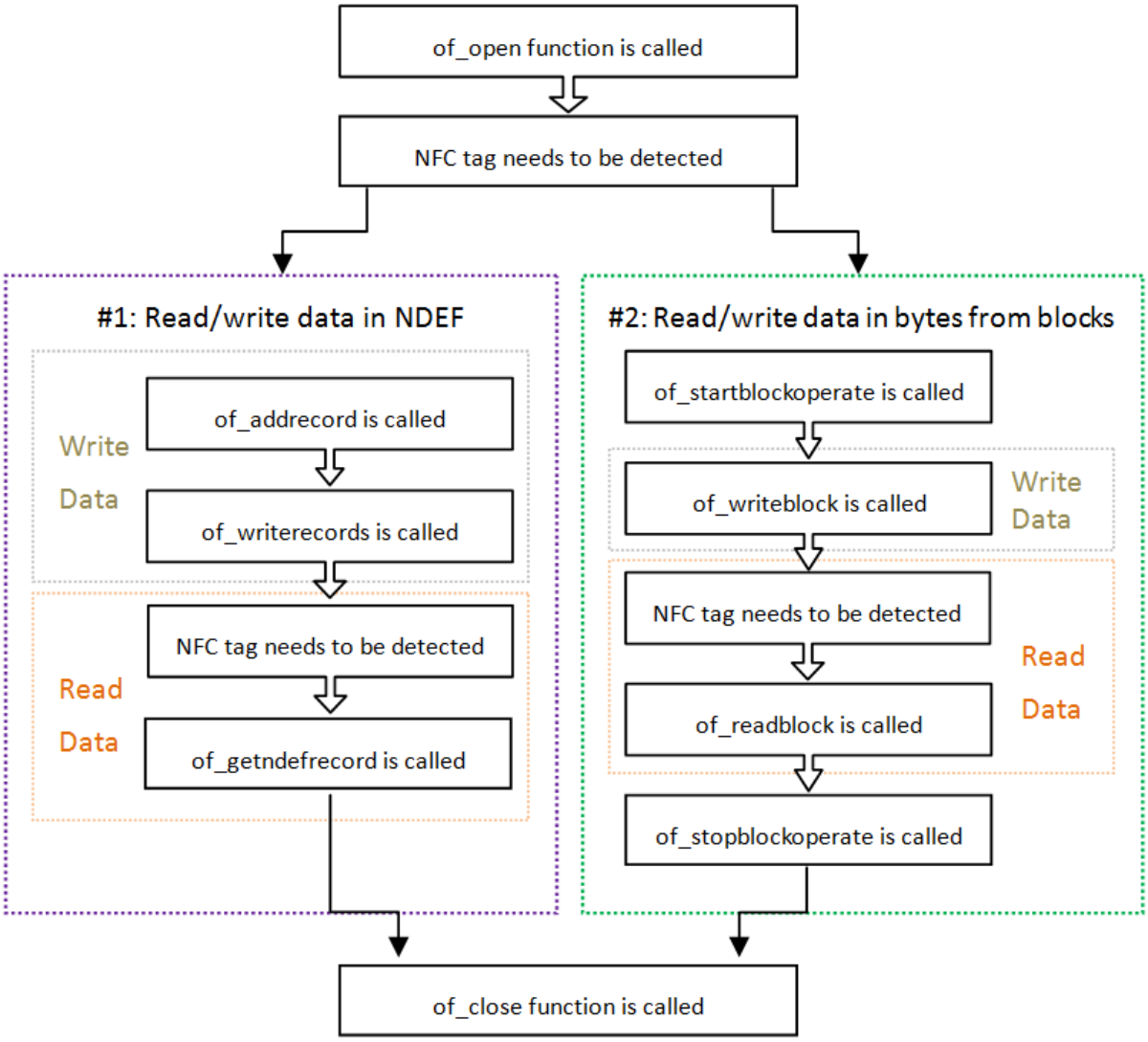
Type	Variable Name	Description
int	i_messagetype	0 - URL message 1 - Mime message 2 - App message 3 - Text message 4 - Poster message

Type	Variable Name	Description
		5 - Alternative carrier message 6 - Handover carrier message 7 - Handover request 8 - Handover select 9 - External message
string	`s_Domain`	This parameter is required when `i_messagetype` is External message.
string	`s_Type`	Data type or MIME type. This parameter is required when `i_messagetype` is Mime message or External message.
string	`s_Payload`	Valid data.

2.4.11.1.5 Code example

Below flowchart illustrates how NFC functionality works.

Figure 2.11: NFC flowchart



- of_open: it must be called before the NFC tag is detected, otherwise, the other functions (such as the read/write functions) will not take effect. If the NFC tag is detected before this function is called, the tag will need to be detected again after this function is called.
- of_addrecord and of_writerecords: of_addrecord should be called first to add one or more records, then of_writerecords should be called to write all records at one time.
- of_getndefrecord: After the records are successfully written to the tag, they cannot be read immediately, the tag will need to be detected by the reader again, and then the records can be read successfully by the of_getndefrecord function.
- of_startblockoperate: it must be called before of_writeblock or of_readblock is called, otherwise of_writeblock and of_readblock will not take effect.
- of_readblock: After the records are successfully written to the tag, they cannot be read immediately, the tag will need to be detected by the reader again, and then the records can be read successfully by the of_readblock function.

Code example for method 1: Read/write data in NDEF (NFC Data Exchange Format)

```
eon_mobile_nfcex   Inv_mobile_nfc  //create nfc object
eon_mobile_str_nfcrecord      istr_nfcrecord

//open nfc object
Inv_mobile_nfc.of_open()

//read nfc tag record, li_index is the index of the block which you want to get.
Inv_mobile_nfc.of_getNdefRecord(li_index,istr_nfcrecord)

//write nfc tag record
Inv_mobile_nfc.of_addrecord(istr_nfcrecord)
Inv_mobile_nfc.of_writeRecords()

//close nfc object
Inv_mobile_nfc.of_close()
```

Code example for method 2: Read/write data in bytes from blocks

```
eon_mobile_nfcex   Inv_mobile_nfc  //create nfc object
eon_mobile_str_nfcrecord      istr_nfcrecord

//open nfc object
Inv_mobile_nfc.of_open()

//if necessary
Inv_mobile_nfc.of_setTimeout(li_timeout)
Inv_mobile_nfc.of_setkey(ls_key)

Inv_mobile_nfc.of_startBlockOperate()

//read nfc tag record
Inv_mobile_nfc.of_readBlock(li_block,ibl_data)

//write nfc tag record
Inv_mobile_nfc.of_writeBlock(li_block,ibl_data)

Inv_mobile_nfc.of_stopBlockOperate()

//close nfc object
Inv_mobile_nfc.of_close()
```

2.4.12 Notification

Sends a notification to the notification bar when the mobile application is running in the foreground. If the application is closed or running in the background, no notification can be sent.

Usage example

- Airlines:

  Because of the weather effects and other force majeures, the airlines can send such warm notifications to inform passengers of the latest flight dynamics, so as to reassure them and reduce some troublesome cases.

- Banking and finance:

  The e-notification service is a very practical solution that allows quick and safe transmission of any personalized message via electronic mail or SMS. It is especially helpful as a form of automatic notification of, for example, change in account balance, issuance of an invoice, deadline for installment payment, etc. The client is provided with the latest information and always stays updated.

- Healthcare:

  Reminding patients of their dentist appointments is an important part of any dentistry which helps build and strengthen the relationship between patient and dentist. When patients receive those reminders, they are not only reminded of their appointment they are also made aware of the quality and care of the business.

2.4.12.1 eon_mobile_notificationex object

2.4.12.1.1 Functions

of_addmessage

Description

Sends a notification to the notification bar when the mobile application is running in the foreground. If the application is closed or running in the background, no notification can be sent.

Supported on mobile client only.

Syntax 1

notification.of_addmessage (value string *as_message*)

Syntax 2

notification.of_addmessage (value string *as_title*, value string *as_message*)

Argument Type	Argument	Description
eon_mobile_notificationex	*notification*	The name of the eon_mobile_notificationex object.

Argument Type	Argument	Description
string	*as_title*	The title to be displayed on the notification bar. This parameter takes effect on the Android device only. On the iOS device, this parameter will be ignored, and the title is always the application name, which means if your app runs in Appeon Workspace, the title will be "AppeonMobile"; if your app is a native app packaged via the PowerServer Toolkit Package tool, the title will be whatever specified in the App Name field when packaged.
string	*as_message*	The message to be displayed on the notification bar.

Return value

Long.

>0 - Return the current notification handle if sending notification successfully. of_removemessage function can use this handle to delete the notification.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

Code example

```
eon_mobile_notificationex ieon_notification
integer ii_notification

ieon_notification = CREATE eon_mobile_notificationex

string ls_now
ls_now = string (now(), 'HH:MM:SS')
ii_notification = ieon_notification.of_addmessage ("Time", ls_now)
```

Note that the of_addmessage function will send a notification to the notification bar; and the notification will show on top of the screen when the end user swipe down from the top of the screen to trigger the Notification Center.

of_removemessage

Description

Removes a notification from the notification bar.

Supported on mobile client only.

Syntax

notification.of_removemessage (value long *al_handle*)

Argument Type	Argument	Description
eon_mobile_notificationex	*notification*	The name of the eon_mobile_notificationex object.
long	*al_handle*	Specifies the handle of the notification, which is returned by the of_addmessage() function.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

Code example

```
eon_mobile_notificationex ieon_notification
integer ii_notification

ieon_notification = CREATE eon_mobile_notificationex

string ls_now
ls_now = string (now(), 'HH:MM:SS')
ii_notification = ieon_notification.of_addmessage ("Time", ls_now) // add
ieon_notification.of_removemessage (ii_notification) // remove
```

Do not use the of_removemessage function to remove the notification from the notification bar as soon as it is added, otherwise it will not show in the Notification Center. Instead, you should remove the notification from the Notification Center as needed.

2.4.13 Offline

Checks and applies the updates of the current application.

2.4.13.1 eon_offlineex object

2.4.13.1.1 Functions

of_applydbupdate

Description

Updates the local database files of the current application.

Supported on mobile client only.

Syntax

offline.of_applydbupdate()

Argument Type	Argument	Description
eon_offlineex	*offline*	The name of the eon_offlineex object.

Return value

Integer.

0 - The DB structure of the current application is the same as that in the server, and no need to update.

1 - Success.

-1 - Failure or it is called in PowerBuilder.

-2 - It is called in PowerServer Web and this operation is not supported.

of_checkappeonserver

Description

Detects if PowerServer exists.

Supported on mobile client only.

Syntax

`offline.of_checkappeonserver()`

Argument Type	Argument	Description
eon_offlineex	*offline*	The name of the eon_offlineex object.

Return value

Integer.

1 - PowerServer exists.

-1 - PowerServer does not exist or there is an error in the checking of the PowerServer, or it is called in PowerBuilder.

of_applyupdate

Syntax 1

Syntax 2

Description 1

Updates the current application from the server.

Supported on mobile client only.

Syntax 1

`offline.of_applyupdate ( boolean ab_showdialog )`

Argument Type	Argument	Description	
eon_offlineex	*offline*	The name of the eon_offlineex object.	
boolean	*ab_showdialog*	This argument controls whether to	

Argument Type	Argument	Description	
		display a dialog box that prompts the end user to restart the app after updated and confirm to install the update. The prompt message is "After the app is updated, you need to restart the app to make the new version take effect. Please confirm your update." The end user can choose **OK** to update the app, or choose **Cancel** to not to update the app. True - To display the dialog box. False - Not to display the dialog box.	

Return value 1

Integer.

1 - Success.

2 - User has clicked **Cancel** in the dialog box to not to update the app.

0 - Latest app and no need to update.

-1 - Failure or it is called in PowerBuilder.

-2 - It is called in PowerServer Web and this operation is not supported.

Description 2

Updates the current application from the server according to the return value of of_checkupdate.

Supported on mobile client only.

Syntax 2

```
offline.of_applyupdate ()
```

Argument Type	Argument	Description
eon_offlineex	*offline*	The name of the eon_offlineex object.

Return value 2

Integer.

1 - Success.

0 - Latest app and no need to update.

-1 - Failure or it is called in PowerBuilder.

-2 - It is called in PowerServer Web and this operation is not supported.

Note: The following application configuration information will be synchronized when running the offline mobile application for the first time or when calling the of_applyupdate function.

- AEM configuration
  1. Application > Local Database
  2. Application > PB Features > INI Files
  3. Application > PB Features > Decimal Precision
  4. Application > Client Features > [codeexamples] > Application Title
  5. Application > Client Features > [codeexamples] > Retina Display (Mobile Only)
  6. Application > Data Transfer > Encoding
  7. Mobile UI Resizing
- Application configuration
- Data source
- Embedded SQL
- DW SQL
- Local database files

of_checkupdate

Description

Detects if there is any update for the current application.

Supported on mobile client only.

Syntax

offline.of_checkupdate( )

Argument Type	Argument	Description
eon_offlineex	*offline*	The name of the eon_offlineex object.

Return value

Integer.

0 - There is no update.

1 - There is an update only for the application.

2 - There is an update only for the database file.

3 - There are updates for both the database file and the application.

-1 - Failure in the update check or it is called in PowerBuilder.

-2 - It is called in PowerServer Web and this operation is not supported.

2.4.14 Online Payment

Makes online payment with the third-party payment service providers such as PayPal, AliPay, Western Union, Google etc. Currently PayPal is the only provider supported by Appeon.

2.4.14.1 eon_mobile_paymentex object

2.4.14.1.1 Properties

integer ii_appplatformtype

Platform type. Currently PayPal is the only provider supported by Appeon.

0 - Unknown.

1 - PayPal.

integer ii_errorcode

Error code which is returned when the oe_error event is triggered. It is designed to hold the error code returned from the payment platform, but currently PayPal returns no error code, so this property is reserved for future use.

powerobject ipo_bindevent

The object to bind with the oe_error event.

string is_bindwitherrorevent

The name of the event bound with the oe_error event of the powerobject ipo_bindevent.

string is_errortext

Error message which is returned when the oe_error event is triggered.

string is_paymentcancel

The name of the event bound with the oe_paymentcancel event.

string is_paymentfailed

The name of the event bound with the oe_paymentfailed event.

string is_paymentok

The name of the event bound with the oe_paymentok event.

2.4.14.1.2 Events

oe_error

Description

Occurs when there is any error in this functionality.

Supported on mobile client only.

Syntax

None.

Return value

None.

oe_paymentcancel

Description

Occurs when the payment is cancelled or terminated.

Supported on mobile client only.

Syntax

None.

Return value

None.

oe_paymentfailed

Description

Occurs when the payment failed.

Supported on mobile client only.

Syntax

None.

Return value

None.

oe_paymentok

Description

Occurs when the payment is successful.

Supported on mobile client only.

Syntax

None.

Return value

None.

2.4.14.1.3 Functions

of_getappplatform

Description

Detects which payment platform the application is connecting now to make payments. Currently PayPal is the only supported platform.

Supported on mobile client only.

Syntax

payment.of_getappplatform ()

Argument Type	Argument	Description
eon_mobile_paymentex	*payment*	The name of the eon_mobile_paymentex object.

Return value

Integer.

0 - Unknown.

1 - PayPal.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_init

Description

Initializes the payment record.

Supported on mobile client only.

Syntax

payment.of_init (eon_mobile_str_paymentinit *astr_paymentinit*)

Argument Type	Argument	Description
eon_mobile_paymentex	*payment*	The name of the eon_mobile_paymentex object.
eon_mobile_str_paymentinit	*astr_paymentinit*	The payment record to be initialized. (For details, refer to the property of the

Argument Type	Argument	Description
		eon_mobile_str_paymentinit structure.)

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_register

Description

Registers the object and the event to be bound with the oe_paymentok, oe_paymentcancel, oe_paymentfailed, and oe_error events.

Supported on mobile client only.

Syntax

payment.of_register (powerobject *apb_bind*, string *as_paymentok*, string *as_paymentcancel*, string *as_paymentfailed*, string *as_error*)

Argument Type	Argument	Description
eon_mobile_paymentex	*payment*	The name of the eon_mobile_paymentex object.
powerobject	*apb_bind*	The object to be bound with the oe_paymentok, oe_paymentcancel, oe_paymentfailed, and oe_error events.
string	*as_paymentok*	The event to be bound with the oe_paymentok event.
string	*as_paymentcancel*	The event to be bound with the oe_paymentcancel event.
string	*as_paymentfailed*	The event to be bound with the oe_paymentfailed event.
string	*as_error*	The event to be bound with the oe_error event.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_setappplatform

Description

Sets the payment platform which the application will connect to make payments. Currently PayPal is the only supported platform.

Supported on mobile client only.

Syntax

payment.of_setappplatform (integer *ai_appplatform*)

Argument Type	Argument	Description
eon_mobile_paymentex	*payment*	The name of the eon_mobile_paymentex object.
integer	*ai_appplatform*	Type of the payment platform. Currently PayPal is the only supported platform. 0 - Unknown. 1 - PayPal.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_submit

Description

Submits the payment record.

Supported on mobile client only.

Syntax

payment.of_submit (eon_mobile_str_paymentsubmit *astr_paymentsubmit*)

Argument Type	Argument	Description
eon_mobile_paymentex	*payment*	The name of the eon_mobile_paymentex object.
eon_mobile_str_paymentsubmit	*astr_paymentsubmit*	The payment record to be submitted. (For details, refer to the property of the eon_mobile_str_paymentsubmit structure.)

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_triggerevent

Description

Triggers the object event that is bound with oe_error or the other events.

Supported on mobile client only.

Syntax

payment.of_triggerevent (string *as_event*)

Argument Type	Argument	Description
eon_mobile_paymentex	*payment*	The name of the eon_mobile_paymentex object.
string	*as_event*	The event name bound with oe_error or the other events.

Return value

None.

2.4.14.1.4 Structures

eon_mobile_str_paymentinit

Description

Struct.

Detailed information about the initialized payment record.

Property

Type	Variable Name	Description
string	s_clientid	PayPal ID for the merchant.

eon_mobile_str_paymentsubmit

Description

Struct.

Detailed information about the payment record to be submitted.

Property

Type	Variable Name	Description
double	dbl_countmoney	Total amount of money.
string	s_currency	Currency.

Type	Variable Name	Description
string	s_productname	Product name.

2.4.14.1.5 Code example

Below is the sample code for making an online payment via PayPal.

```
eon_mobile_paymentex   lnv_payment
eon_mobile_str_paymentinit   lstr_paymentinit
eon_mobile_str_paymentsubmit  lstr_paymentsubmit
integer li_re

lnv_payment = create eon_mobile_paymentex

//bind oe_paymentok, oe_paymentcancel, oe_paymentfailed, and oe_error events of
 lnv_payment with
//ue_paymentok, ue_paymentcancel, ue_paymentfailed, and ue_error events of the
 parent window.
lnv_payment.of_register ( parent, "ue_paymentok", "ue_paymentcancel",
 "ue_paymentfailed", "ue_error")

//initialize platform
lnv_payment.of_setappplatform(1)

//initialize payment
lstr_paymentinit.s_clientid = "AXMNlBBgmfChHPgrMAtrrdTtkWk52THb9Hl54uZ6vASDWSOll8"
li_re = lnv_payment.of_init (lstr_paymentinit)

if li_re =  1 then
         //submit payment
         lstr_paymentsubmit.dbl_countmoney = 9.99
         lstr_paymentsubmit.s_currency = "USD"
         lstr_paymentsubmit.s_productname = "shoes"

         li_re = lnv_payment.of_submit (lstr_paymentsubmit)
         if li_re = 1 then
                  MessageBox ("","Payment is successful!")
         else
                  MessageBox ("","Payment failed!")
         end if
else
         MessageBox("","Failed to initialize payment!")
end if

destroy lnv_payment
```

2.4.15 Textchecker

Checks spelling in a text field.

2.4.15.1 eon_mobile_textcheckex object

2.4.15.1.1 Functions

of_completionsforpartialword

Description

Returns a list of words that are possible completions for a partially entered word.

Supported on mobile client only.

Syntax

textchecker.of_completionsforpartialword (value eon_mobile_str_textcheckoption *astr_textcheckoption*, ref string *as_completions[]*)

Argument Type	Argument	Description
eon_mobile_textcheckerex	*textchecker*	The name of the eon_mobile_textcheckerex object.
eon_mobile_str_textcheckoption	*astr_textcheckoption*	Check option. See variable list of eon_mobile_str_textcheckoption.
string	*as_completions[]*	Returns a list of words, each of which is a completion of a partially entered word represented by range in string. If no possible completions are found, the method returns an empty array.

Return value

Integer.

1 - Found possible completions.

0 - Found no possible completions.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getignoredwords

Description

Returns the words that the receiver ignores when spell-checking.

The spell checker excludes ignored words as misspelled words during the current spell-checking session only.

Supported on mobile client only.

Syntax

textchecker.of_getignoredwords (ref string *as_words[]*)

Argument Type	Argument	Description
eon_mobile_textcheckerex	*textchecker*	The name of the eon_mobile_textcheckerex object.
string	*as_words[]*	Returns an array of strings, each of which specifies a word the receiver ignores.

Return value

Integer.

1 - Found any ignored words.

0 - Found no ignored words.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_setignoredwords

Description

Sets the list of words that the receiver should ignore, and the ignored words are not treated as misspelled words.

Supported on mobile client only.

Syntax

textchecker.of_setignoredwords (value string *as_words[]*)

Argument Type	Argument	Description
eon_mobile_textcheckerex	*textchecker*	The name of the eon_mobile_textcheckerex object.
string	*as_words[]*	Sets an array of strings, each of which specifies a word the receiver should ignore.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getlanguages

Description

Gets the languages for which the text checker can perform spell-checking.

Supported on mobile client only.

Syntax

textchecker.of_getlanguages (ref string *as_languages[]*)

Argument Type	Argument	Description
eon_mobile_textcheckerex	*textchecker*	The name of the eon_mobile_textcheckerex object.
string	*as_languages[]*	Returns an array of strings representing ISO 639-1

Argument Type	Argument	Description
		language codes or combined ISO 639-1 language codes and ISO 3166-1 regional codes (for example, en_US).

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_getmisspelledword

Description

Gets the range of the first misspelled word encountered.

Supported on mobile client only.

Syntax

textchecker.of_getmisspelledword (value eon_mobile_str_textcheckoption *astr_textcheckoption*, ref long *al_start*, ref long *al_length*)

Argument Type	Argument	Description
eon_mobile_textcheckerex	*textchecker*	The name of the eon_mobile_textcheckerex object.
eon_mobile_str_textcheckoption	*astr_textcheckoption*	Check option. See variable list of eon_mobile_str_textcheckoption.
long	*al_start*	Returns a long whose value is the starting position of the first misspelled word.
long	*al_length*	Returns a long whose value is the length of the first misspelled word.

Return value

Integer.

1 - Found the misspelled word.

0 - Found no misspelled words.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_guessesforword

Description

Returns a list of words that are possible valid replacements for a misspelled word.

Supported on mobile client only.

Syntax

textchecker.of_guessesforword (value eon_mobile_str_textcheckoption *astr_textcheckoption*, ref string *as_guesses[]*)

Argument Type	Argument	Description
eon_mobile_textcheckerex	*textchecker*	The name of the eon_mobile_textcheckerex object.
eon_mobile_str_textcheckoption	*astr_textcheckoption*	Check option. See variable list of eon_mobile_str_textcheckoption.
string	*as_guesses[]*	Returns an array of strings each of which might be a correct substitute (that is, a guess) for a misspelled word in the given range of the string. If no possible guesses are found, the method returns an empty array.

Return value

Integer.

1 - Found possible guesses.

0 - Found no possible guesses.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_haslearnedword

Description

Detects if the text checker has learned the specified word.

Supported on mobile client only.

Syntax

textchecker.of_haslearnedword (value string *as_word*)

Argument Type	Argument	Description
eon_mobile_textcheckerex	*textchecker*	The name of the eon_mobile_textcheckerex object.

Argument Type	Argument	Description
string	*as_word*	A string representing a word.

Return value

Integer.

1 - The text check has learned the word.

0 - The text check hasn't learned the word.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_learnword

Description

Tells the text checker to learn the specified word so that it is not evaluated as misspelled.

Supported on mobile client only.

Syntax

textchecker.of_learnword (value string *as_word*)

Argument Type	Argument	Description
eon_mobile_textcheckerex	*textchecker*	The name of the eon_mobile_textcheckerex object.
string	*as_word*	A string representing the word for the text checker to learn.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_unlearnword

Description

Tells the text checker to unlearn the specified word.

Supported on mobile client only.

Syntax

textchecker.of_unlearnword (value string *as_word*)

Argument Type	Argument	Description
eon_mobile_textcheckerex	*textchecker*	The name of the eon_mobile_textcheckerex object.

Argument Type	Argument	Description
string	*as_word*	A string representing the word for the text checker to unlearn.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

2.4.15.1.2 Structures

eon_mobile_str_textcheckoption

Description

Struct.

Checks spelling.

Property

Type	Variable Name	Description
string	s_source	The string which you want to check.
long	l_start	A long indicating where the check will begin in s_source.
long	l_length	Starting from the position specified by l_start, the number of characters needed to be checked.
string	s_language	The language of the words to be checked for correct spelling. This variable takes no effect on the Android OS if the text is input in multiple languages.

2.4.15.1.3 Code example

The following script is in the EditChanged event for a SingleLineEdit control, used to detect if the input word is correct.

```
long ll_start,ll_length,ll_return1,ll_return2
string ls_return[],ls_data
eon_mobile_str_textcheckoption  leon_str_textcheck
str_textcheck  lstr_textcheck
```

```
leon_str_textcheck.s_language = 'en_US'
leon_str_textcheck.s_source = this.text
leon_str_textcheck.l_start = 1
leon_str_textcheck.l_length = len(this.text)
ll_return1 = ieon_textchecker.of_getmisspelledword ( leon_str_textcheck, ll_start,
 ll_length) //returns the starting position and length of the first misspelled word
if ll_return1 = 1 then
   leon_str_textcheck.l_start = ll_start
   leon_str_textcheck.l_length = ll_length //IMPORTANT: passes the length to the
 structure so it can exactly obtain the misspelled word
   if cbx_1.checked then
       ll_return2 = ieon_textchecker.of_guessesforword ( leon_str_textcheck,
 ls_return) //returns a list of possible guesses
   else
       ll_return2 = ieon_textchecker.of_completionsforpartialword
 ( leon_str_textcheck, ls_return) //returns a list of possible completions
   end if
   if upperbound(ls_return) > 0 then
       lstr_textcheck.s_old = mid(this.text,ll_start,ll_length)
       lstr_textcheck.s_new = ls_return
       lstr_textcheck.l_x = parent.pointerx( )
       lstr_textcheck.l_y = parent.pointery( )
       openwithparm(w_rightword,lstr_textcheck)
       ls_data = message.stringparm
       if len(ls_data) > 0 then
           this.text = replace(this.text,ll_start,ll_length,ls_data)
       end if
   end if
end if
```

2.4.16 WebView

Loads the HTML page and/or runs JavaScript, displays files such as text file, XML file, PDF etc. and explores the Web page.

2.4.16.1 eon_mobile_webviewex object

2.4.16.1.1 Properties

integer ii_errorcode

Error code which is returned when the oe_error event is triggered.

string is_errortext

Error message which is returned when the oe_error event is triggered.

2.4.16.1.2 Events

oe_error

Description

Occurs when there is any error in this functionality. In the Android OS, when this event is triggered, an error page from the Android OS will be displayed. However, unlike Android, iOS will not return any error page or error message box, therefore, it is recommended that you write code in this event to display an error message box or an error page which you specially prepared for the iOS device.

Supported on mobile client only.

Syntax

None.

Return value

None.

oe_jscallbackappeon

Description

Occurs automatically when the Web page loaded into the eon_mobile_webviewex object calls this JavaScript function: `callbackappeon` (*`as_data`*) for iOS, or eon_android.`callbackappeon` (*`as_data`*) for Android. So this event and the JavaScript function together can be utilized as communication bridge between the Web page and the Appeon mobile app.

`callbackappeon` (*`as_data`*) is a JavaScript function predefined in Appeon for use in iOS only, and the eon_android JavaScript object and its `callbackappeon` (*`as_data`*) function are predefined in Appeon for use in Android only. Users need not to declare them before calling them; and users should prevent declaring and overloading them too. If these functions are called by a Web page that is not loaded into the eon_mobile_webviewex object, they will return errors such as "'callbackappeon' is undefined" or "'eon_android' is undefined".

Supported on mobile client only.

Syntax

`oe_jscallbackappeon` (string *`as_data`*)

Argument Type	Argument	Description
string	*`as_data`*	Receives the string passed in by the parameter of the `callbackappeon` JavaScript function. You will need to make sure the value to be passed in is a string, not of any other data type.

Return value

None.

Code example

The following code shows how to encapsulate a `callbackappeon` (*`as_data`*) function and then call it instead of calling the `callbackappeon` (*`as_data`*) or eon_android.`callbackappeon` (*`as_data`*) directly.

```
function jscallbackappeon(as_data)
 {
         if ( typeof(callbackappeon) == "function" )
         {
                  callbackappeon(as_data);
         }
         else if (typeof(eon_android) == "object" )
         {
```

```
                    eon_android.callbackappeon(as_data);
        }

}
```

oe_loadend

Description

Occurs when the of_startloading function finishes loading the page.

Supported on mobile client only.

Syntax

None.

Return value

None.

oe_loadstart

Description

Occurs when the of_startloading function starts loading the page.

Supported on mobile client only.

Syntax

None.

Return value

None.

2.4.16.1.3 Functions

of_cangoback

Description

Indicates whether the user can move backward.

Supported on mobile client only.

Syntax

webview.of_cangoback ( )

Argument Type	Argument	Description
eon_mobile_webviewex	*webview*	The name of the eon_mobile_webviewex object.

Return value

Integer.

0 - Unable to move backward.

1 - Able to move backward.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_cangoforward

Description

Indicates whether the user can move forward.

Supported on mobile client only.

Syntax

webview.of_cangoforward( )

Argument Type	Argument	Description
eon_mobile_webviewex	*webviewx*	The name of the eon_mobile_webviewex object.

Return value

Integer.

0 - Unable to move forward.

1 - Able to move forward.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_goback

Description

Loads the previous location in the back-forward list.

Supported on mobile client only.

Syntax

webview.of_goback( )

Argument Type	Argument	Description
eon_mobile_webviewex	*webviewx*	The name of the eon_mobile_webviewex object.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_goforward

Description

Loads the next location in the back-forward list.

Supported on mobile client only.

Syntax

webview.of_goforward()

Argument Type	Argument	Description
eon_mobile_webviewex	*webview*	The name of the eon_mobile_webviewex object.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_loadhtmlstring

Description

Loads the string of the HTML page. Only HTM or HTML file is supported.

Supported on mobile client only.

Syntax

webview.of_loadhtmlstring (value string *as_htmlstring*)

Argument Type	Argument	Description
eon_mobile_webviewex	*webview*	The name of the eon_mobile_webviewex object.
string	*as_htmlstring*	The string of the HTML page which you want to load.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

Code example

```
string ls_filepath,ls_file
integer li_ret
long ll_file

ls_file = appeongetcachedir()+"/plugin/customer.html"

ll_file = fileopen(ls_file,textmode!,read!,shared!)
filereadex(ll_file,ls_filepath)
messagebox('',ls_filepath)
```

```
li_ret = uo_webview.of_loadhtmlstring (ls_filepath)
if li_ret > -1 then
 messagebox("note:", "success of_loadhtmlstring: Filepath#"  +ls_filepath + ",
 return value#" +  string(li_ret))
else
 messagebox("note:", "failed of_readBlock:: Filepath#"  +ls_filepath + ", return
 value#" +  string(li_ret))
end if

fileclose(ll_file)
```

of_loadlocalfile

Description

Loads and displays the local file.

In iOS, the following file types can be loaded and displayed: PDF, TXT, XML, Word, Excel, and PPT. Android supports fewer file types as it can only support text types such as TXT, XML, and HTML. Android does not support loading the PDF file, therefore you will not be able to load the local PDF file in Android, but if you put the PDF file on a Web site accessible via Internet, and allow the mobile device to connect with the Web site and Google Docs (docs.google.com), then Appeon can parse the PDF file online via Google Docs (docs.google.com) and then load the PDF file into the Android device, for example, if you put the PDF file online as *http://demo.appeon.com/AEM/appeon_mobile_2013R2_datasheet.pdf*, then Appeon mobile app will automatically display the online PDF file via URL *http://docs.google.com/gview?embedded=true&url=http://demo.appeon.com/AEM/appeon_mobile_2013R2_datasheet.pdf*.

Supported on mobile client only.

Syntax

webview.of_loadlocalfile (value string *as_filepath*)

Argument Type	Argument	Description
eon_mobile_webviewex	*webview*	The name of the eon_mobile_webviewex object.
string	*as_filepath*	The path of the local file which you want to load.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

Code example

```
int  li_ret
string ls_filepath

ls_filepath = appeongetcachedir()+"/plugin/pig.txt"
li_ret = uo_webview.of_loadlocalfile (ls_filepath)
if li_ret > -1 then
```

```
 messagebox("note:", "success of_loadlocalfile:FilePath:" +ls_filepath + ", return
 value#" +  string(li_ret))
else
 messagebox("note:", "failed of_loadlocalfile:FilePath:" +ls_filepath + ", return
 value#" +  string(li_ret))
end if
```

of_reload

Description

Reloads the current page.

Supported on mobile client only.

Syntax

webview.of_reload ( )

Argument Type	Argument	Description
eon_mobile_webviewex	*webview*	The name of the eon_mobile_webviewex object.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

of_runjavascript

Description

Runs the JavaScript code snippet.

Supported on mobile client only.

Syntax

webview.of_runjavascript (value string *as_strjs*, ref string *as_strret*)

Argument Type	Argument	Description
eon_mobile_webviewex	*webview*	The name of the eon_mobile_webviewex object.
string	*as_strjs*	The script which you want to run. You can use the syntax "functionname()" to call the JavaScript methods or the methods of the current page.
string	*as_strret*	The result of running script. Any value returned from JavaScript will be returned

Argument Type	Argument	Description
		here; if there is no return value from JavaScript (like Alert, or simple script for assigning values to variables) or if there is any error running JavaScript, then an empty string will be returned.

Return value

Integer.

1 - If the *as_strret* parameter returns a non-empty string.

-1 - If the *as_strret* parameter returns an empty string, or if it is called in PowerBuilder or PowerServer Web.

Code example

```
string  ls_strjs,ls_strret
integer li_ret
ls_strjs= 'javascript:alert("test")'

li_ret = uo_webview.of_runjavascript (ls_strjs,ls_strret)

if li_ret > -1 then
 messagebox("note:", "success of_runjavascript:" +ls_strret+ ",return value#" +
 string(li_ret))
 st_21.text = ls_strret
else
 messagebox("note:", "failed of_runjavascript:" +ls_strret+ ", return value#" +
 string(li_ret))
 st_21.text = ""
end if
```

of_startloading

Description

Starts the loading of the page.

Supported on mobile client only.

Syntax

webview.of_startloading (value string *as_url*)

Argument Type	Argument	Description
eon_mobile_webviewex	*webview*	The name of the eon_mobile_webviewex object.
string	*as_url*	The URL of the page which you want to load.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

Code example

```
int  li_ret
string ls_url

ls_url= "www.google.com"
li_ret = uo_webview.of_startloading(ls_url)
if li_ret > -1 then
 messagebox("note:", "success of_startloading:URL:" + ls_url + "#return value#"
 +string(li_ret))
else
 messagebox("note:", "failed of_startloading:URL:" + ls_url + "#return value#"
 +string(li_ret))
end if
```

of_stoploading

Description

Stops the loading of the current page.

Supported on mobile client only.

Syntax

webview.of_stoploading()

Argument Type	Argument	Description
eon_mobile_webviewex	*webview*	The name of the eon_mobile_webviewex object.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

3 Appeon Resize PBL Reference

The Appeon Resize PBL (appeon_resize.pbl) has been removed from the product, and the object under the Appeon Resize PBL has been moved to the Appeon Workarounds PBL (appeon_workarounds.pbl). For the syntax and code examples of the Appeon resize object, refer to the Appeon Resize Object section.

4 Integrate with Cordova Plugins

4.1 Overview

Apache Cordova plugins provide JavaScript interfaces that can be called to use native device capabilities to enhance the features and functionality of a mobile application. To enable Appeon mobile apps to take advantage of Cordova plugins, PowerServer Mobile has packaged the common Cordova plugins and provided two solutions for a PowerBuilder developer to call the plugins:

- Solution 1: Call the PowerScript interfaces that Appeon Cordova PBL provides. The Appeon Cordova PBL provides interfaces that can automatically execute the corresponding JavaScript methods and return the results from JavaScript, therefore, the developer is able to call the Cordova plugins as easily as call a PowerBuilder object, without needing to know the JavaScript language.

  For detailed instructions, refer to Appeon Cordova PBL.

  However, the Appeon Cordova PBL only encapsulates the following Cordova plugins as PowerBuilder objects: Bluetooth, Bluetooth Low Energy, Contacts, Fingerprint authentication, Geolocation, Device orientation, & Vibration.

- Solution 2: Call the JavaScript interfaces that Cordova plugins provide. This requires the developer to at least be able to read the JavaScript code in the Cordova plugins. This solution gives the developer most flexibility to use any Cordova plugins that are included in the Appeon product (view list). If you found a Cordova plugin not included in the Appeon product, you can contact Appeon Support (support@appeon.com) to make plans for including it.

  For detailed instructions, refer to Cordova plugins.

If you found any issue with the Cordova plugins, please first search the Cordova plugin issue list to see if the issue already exists; if the issue does not exist, you can submit the issue after logged in.

4.2 Appeon Cordova PBL (solution 1 [211])

4.2.1 Introduction to Appeon Cordova PBL

Appeon Cordova PBL encapsulates the following Cordova plugins as PowerBuilder objects. Though not all of the current Cordova plugins are encapsulated in the Appeon Cordova PBL, you may still find it useful because the most frequently-used ones are in it and they can be called as easily as a PowerBuilder object.

- Bluetooth (https://github.com/tanelih/phonegap-bluetooth-plugin)
- Bluetooth Low Energy (https://www.npmjs.com/package/cordova-plugin-bluetoothle)
- Contacts (https://www.npmjs.com/package/cordova-plugin-contacts)
- Fingerprint authentication (Android: https://www.npmjs.com/package/cordova-plugin-android-fingerprint-auth, iOS: https://www.npmjs.com/package/cordova-plugin-touch-id)

- Geolocation (https://www.npmjs.com/package/cordova-plugin-mauron85-background-geolocation)
- Device orientation (https://www.npmjs.com/package/cordova-plugin-device-orientation)
- Vibration (https://www.npmjs.com/package/cordova-plugin-vibration)

Appeon Cordova PBL is named **appeon_cordova.pbl**, located in the *developTempFile\appeon_workarounds* folder under the PowerServer installation directory. For example, *C:\inetpub\wwwroot\appeon\developTempFile\appeon_workarounds*.

To use the objects provided in the Appeon Cordova PBL, you will need to add the **appeon_cordova.pbl** and **appeon_workarounds.pbl** library to the **Library Search Path** of the application.

4.2.2 eon_cordova_base object

The eon_cordova_base object defines the basic and common functions/events that will be called by the other objects.

4.2.2.1 Functions

4.2.2.1.1 of_getlasterror

Description

Gets the error information of the last function call.

Syntax

```
of_getlasterror()
```

Return value

String. The error information of the last call.

4.2.2.1.2 of_getlastreturn

Description

Gets the return value of the last function call if that call returns any value.

Syntax

```
of_getlastreturn()
```

Return value

String. The return value of the last call.

4.2.2.1.3 of_getvaluefromkey

Description

Gets the value from a JSON-format string for the specified key name.

Syntax

```
of_getvaluefromkey ( string as_json, string as_key )
```

Parameter

as_json - A JSON-format string.

as_key - Key name. It is case sensitive.

Return value

String. Key value of the specified key.

4.2.2.1.4 of_init

Description

Connects with the Cordova plugin, detects if the plugin is available to call, and binds the current PowerBuilder object with the JavaScript object defined by the plugin. After that, the PowerBuilder object has all of the methods and properties that the JavaScript object has.

Syntax

```
of_init ( )
of_init ( string as_pluginname )
```

Parameter

as_pluginname - Name of the JavaScript object defined by the plugin.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

4.2.2.1.5 of_isios

Description

Detects if the current platform is iOS or Android.

Syntax

```
of_isios ( )
```

Return value

Boolean.

True - iOS

False - Android

Null - It is called in PowerBuilder or PowerServer Web, or there is an error.

4.2.2.1.6 of_register

Description

Registers the object and the event to be bound with the oe_success event and the oe_error event.

Syntax

`of_register` (powerobject *`apb_bind`*, string *`as_successevent`*, string *`as_errorevent`*)

Parameter

`apb_bind` - The object to be bound with the oe_error and oe_success event.

`as_successevent` - The event to be bound with the oe_success event.

`as_errorevent` - The event to be bound with the oe_error event.

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

4.2.2.1.7 of_settimeout

Description

Sets the timeout value for the function call.

Syntax

`of_settimeout` (long *`al_timeout`*)

Parameter

`al_timeout` - Milliseconds for the timeout value.

Return value

None

4.2.2.2 Events

4.2.2.2.1 oe_error

Description

Occurs when an error occurs during the function call. If the JavaScript function in the Cordova plugin returns any error, that error will be passed to the *`as_error`* parameter.

Syntax

`oe_error` (string *`as_error`*)

Parameter

`as_error` - A JSON-format string which contains error information returned by the function.

Return value

None

4.2.2.2.2 oe_execjserror

Description

Occurs when there is a syntax error during the function call. However, this event cannot capture all of the errors.

Syntax

`oe_execjserror` (string *`as_error`*)

Parameter

`as_error` - The error information of the last function call. It can also be obtained via of_getlasterror function.

Return value

None

4.2.2.2.3 oe_success

Description

Occurs when the function is successfully executed. If the JavaScript function in the Cordova plugin returns any value, that value will be passed to the *`as_message`* parameter.

Syntax

`oe_success` (string *`as_message`*)

Parameter

`as_message` - A JSON-format string which contains values returned by the function.

Return value

None

4.2.2.3 Properties

oleobject ieon_ole

PowerBuilder OLEObject object to be connected with the Cordova plugin.

powerobject ipo_bindevent

PowerBuilder object to be bound with the JavaScript object.

string is_errorText

Stores the error information returned by the JavaScript function when execution failed.

string is_successText

Stores the value returned by the JavaScript function when execution is successful.

string is_jserrorText

Stores the JavaScript error information when the JavaScript function fails.

string is_successevent

Stores the success event name of the PowerBuilder object.

string is_errorevent

Stores the error event name of the PowerBuilder object.

4.2.3 eon_cordova_bluetooth object (Android only)

This object enables communication between a mobile device (currently only Android-based device) and the bluetooth devices.

4.2.3.1 Functions

4.2.3.1.1 of_disable

Description

Turns off bluetooth on the device. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Syntax

```
of_disable()
```

Return value

None

4.2.3.1.2 of_enable

Description

Turns on bluetooth on the device. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Syntax

```
of_enable()
```

Return value

None

4.2.3.1.3 of_init

Description

Connects with the Cordova bluetooth plugin, detects if the plugin is available to call, and binds the current PowerBuilder object with the JavaScript object defined by the plugin. After that, the PowerBuilder object has all of the methods and properties that the JavaScript object has.

Syntax

```
of_init()
```

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

4.2.3.1.4 of_isenabled

Description

Detects if bluetooth is turned on or not on the device.

Syntax

`of_isenabled( )`

Return value

Boolean

True - Bluetooth is turned on.

False - Bluetooth is turned off.

4.2.3.1.5 of_pair

Description

Pairs your device with a bluetooth device. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Syntax

`of_pair` (string *`as_address`*)

Parameter

`as_address` - The address of the bluetooth device you want to pair with your device.

Return value

None

4.2.3.1.6 of_startdiscovery

Description

Scans for the available bluetooth devices. When successful, the oe_success event will be triggered and the found device(s) will be passed to the event; when failed, the oe_error event will be triggered.

Syntax

`of_startdiscovery( )`

Return value

None

4.2.3.1.7 of_unpair

Description

Unpairs a bluetooth device with your device. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Syntax

`of_unpair` (string *`as_address`*)

Parameter

`as_address` - The address of the bluetooth device you want to unpair with your device.

Return value

None

4.2.3.2 Events

4.2.3.2.1 oe_finish

Description

Occurs when scanning for bluetooth devices is successful. It can be used to scan for bluetooth devices in a loop.

Syntax

`oe_finish` (string *`as_message`*)

Parameter

`as_message` - A JSON-format string which contains values returned by the function such as the address and name of the found bluetooth device.

Return value

None

4.2.3.2.2 oe_success

Description

Occurs when the function is successfully executed. If the JavaScript function in the Cordova plugin returns any value, that value will be passed to the *`as_message`* parameter.

Syntax

`oe_success` (string *`as_message`*)

Parameter

`as_message` - A JSON-format string which contains values returned by the function such as the address and name of the found bluetooth device.

Return value

None

4.2.3.3 Properties

boolean ib_enabled

Indicates whether bluetooth is turned on.

boolean ib_exist

Indicates whether the device exists in the search list.

boolean ib_search

Indicates whether search is successful.

eon_str_cordova_bluetooth ieon_str_bluetooth[]

The structure for storing the address and name of the found bluetooth device.

long il_type

Functionality extension for developers: 1 - isenabled 2 - search.

oleobject ieon_ole

PowerBuilder OLEObject object to be connected with the Cordova plugin.

powerobject ipo_bindevent

PowerBuilder object to be bound with the JavaScript object.

string is_errorevent

Stores the error event name of the PowerBuilder object.

string is_errorText

Stores the error information returned by the JavaScript function when execution failed.

string is_jserrorText

Stores the JavaScript error information when JavaScript call fails.

string is_successevent

Stores the success event name of the PowerBuilder object.

string is_successText

Stores the value returned by the JavaScript function when execution is successful.

4.2.4 eon_cordova_bluetoothle object

This object allows you to interact with Bluetooth Low Energy devices on Android and iOS mobile devices.

4.2.4.1 Functions

4.2.4.1.1 of_base64tojson

Description

Converts a base64 encoded string to a JSON array.

Syntax

`of_base64tojson` (string *`as_data`*)

Parameter

`as_data` - The base64 encoded string.

Return value

String. The decoded data.

4.2.4.1.2 of_characteristics

Description

Discovers the service's characteristics. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Syntax

`of_characteristics` (string *`as_address`*, string *`as_service`*)

Parameter

`as_address` - The address/identifier provided by the scan's return object.

`as_service` - Service UUID.

Return value

None.

4.2.4.1.3 of_close

Description

Closes the connection with a bluetooth LE device. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Syntax

`of_close` (string *`as_address`*)

Parameter

`as_address` - The address/identifier provided by the scan's return object.

Return value

None

4.2.4.1.4 of_close_no_event

Description

Closes the connection with a bluetooth LE device. No oe_success or oe_error event will be triggered.

Syntax

`of_close_no_event` (string *`as_address`*)

Parameter

`as_address` - The address/identifier provided by the scan's return object.

Return value

None

4.2.4.1.5 of_connect

Description

Connects to a bluetooth LE device. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Syntax

`of_connect` (string *`as_address`*)

Parameter

`as_address` - The address/identifier provided by the scan's return object.

Return value

None

4.2.4.1.6 of_decode

Description

Decodes the data after it is read.

Syntax

`of_decode` (string *`as_data`*)

Parameter

`as_data` - The encoded data.

Return value

String. The decoded data.

4.2.4.1.7 of_disconnect

Description

Disconnects from a bluetooth LE device. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Syntax

`of_disconnect` (string *`as_address`*)

Parameter

`as_address` - The address/identifier provided by the scan's return object.

Return value

None

4.2.4.1.8 of_discover

Description

Discovers all the device's services, characteristics, and descriptors after connecting with a device successfully.

When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Supported on Android only. Use of_services and of_characteristics instead on iOS.

Syntax

`of_discover` (string *`as_address`*)

Parameter

`as_address` - The address/identifier provided by the scan's return object.

Return value

None

4.2.4.1.9 of_encode

Description

Encodes the data before it is written to the device.

Syntax

`of_encode` (string *`as_data`*)

Parameter

`as_data` - The decoded data.

Return value

String. The encoded data.

4.2.4.1.10 of_init

Description

Connects with the Cordova bluetooth LE plugin, detects if the plugin is available to call, and binds the current PowerBuilder object with the JavaScript object defined by the plugin. After that, the PowerBuilder object has all of the methods and properties that the JavaScript object has.

Syntax

`of_init` ( )

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

4.2.4.1.11 of_initialize

Description

Initializes bluetooth on the device. This function must be called after of_init and before any others and it should only be called once. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Syntax

`of_initialize` (boolean *`ab_request`*)

Parameter

`ab_request` - Whether to prompt the user to enable bluetooth.

Return value

None

4.2.4.1.12 of_isconnected

Description

Determines whether the device is connected.

Syntax

`of_isconnected()`

Return value

Boolean.

True - Connected.

False - Not connected.

4.2.4.1.13 of_isenabled

Description

Detects if bluetooth is turned on or not on the device. The oe_enable event will be triggered.

Supported on the Android device only.

Syntax

`of_isenabled()`

Return value

None.

4.2.4.1.14 of_read

Description

Reads a particular service's characteristics. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Syntax

`of_read` (string *`as_address`*, string *`as_service`*, string *`as_characteristic`*)

Parameter

`as_address` - The address/identifier provided by the scan's return object.

`as_service` - The service's UUID.

`as_characteristic` - The characteristic's UUID.

Return value

None

4.2.4.1.15 of_services

Description

Discovers the device's services. The UUID of the service can be used by of_characteristics.

When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered; and value returned by the JavaScript function will be passed to parameter of the event.

Supported on iOS only.

Syntax

`of_services` (string *`as_address`*)

Parameter

`as_address` - The address/identifier provided by the scan's return object.

Return value

None

4.2.4.1.16 of_services_no_event

Description

Discovers the device's services. The UUID of the service can be used by of_characteristics.

Supported on iOS only.

Syntax

`of_services_no_event` (string *`as_address`*)

Parameter

`as_address` - The address/identifier provided by the scan's return object.

Return value

String. The service UUID.

4.2.4.1.17 of_startscan

Description

Scans for bluetooth LE devices. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Syntax

`of_startscan` ( )

Return value

None

4.2.4.1.18 of_stopscan

Description

Stops scanning for bluetooth LE devices. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Syntax

`of_stopscan` ( )

Return value

None

4.2.4.1.19 of_write

Description

Writes a particular service's characteristic. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Syntax

`of_write` (string *`as_address`*, string *`as_service`*, string *`as_characteristic`*, string *`as_data`*)

Parameter

`as_address` - The address/identifier provided by the scan's return object.

`as_service` - The service's UUID.

`as_characteristic` - The characteristic's UUID.

`as_data` - Base64 encoded string.

Return value

None

4.2.4.2 Events

4.2.4.2.1 oe_enable

Description

Occurs when the `of_isenabled` function is executed; asks to turn on bluetooth if it is not turned on. If the JavaScript function in the Cordova plugin returns any value, that value will be passed to the *`as_message`* parameter.

Supported on the Android device only.

Syntax

`oe_enable` (string *`as_message`*)

Parameter

`as_message` - A JSON-format string which contains values returned by the function.

Return value

None

4.2.4.2.2 oe_error

Description

Occurs when an error occurs during the function call. If the JavaScript function in the Cordova plugin returns any error, that error will be passed to the *`as_error`* parameter.

Syntax

`oe_error` (string *`as_error`*)

Parameter

`as_error` - A JSON-format string which contains error information returned by the function.

Return value

None

4.2.4.2.3 oe_success

Description

Occurs when the function is successfully executed. If the JavaScript function in the Cordova plugin returns any value, that value will be passed to the *as_message* parameter.

Syntax

oe_success (string *as_message*)

Parameter

as_message - A JSON-format string which contains values returned by the function.

Return value

None

4.2.4.3 Properties

boolean ib_enable

Indicates whether bluetooth is enabled on the device.

boolean ib_exist

Indicates whether the newly scanned device exists in the scan result list.

eon_str_cordova_bluetooth ieon_str_bluetooth[]

The structure for storing the address and name of the found bluetooth device.

long il_type

Functionality extension for developers: 1 - scan, 2 - connect, 3 - read

oleobject ieon_ole

PowerBuilder OLEObject object to be connected with the Cordova plugin.

powerobject ipo_bindevent

PowerBuilder object to be bound with the JavaScript object.

string is_connectinfo

Stores the success or error information when establishing a connection.

string is_errorevent

Stores the error event name of the PowerBuilder object.

string is_errorText

Stores the error information returned by the JavaScript function when execution failed.

string is_jserrorText

Stores the JavaScript error information when JavaScript call fails.

string is_successevent

Stores the success event name of the PowerBuilder object.

string is_successText

Stores the value returned by the JavaScript function when execution is successful.

4.2.4.4 Structures

4.2.4.4.1 eon_str_cordova_bluetooth

Description

Struct.

Stores the information of the bluetooth device including address, name, status etc.

Property

Type	Variable Name	Description
string	`address`	Address of the bluetooth device.
string	`name`	Name of the bluetooth device.
string	`rssi`	RSSI of the bluetooth broadcasting.
string	`advertisement`	Advertisement of the bluetooth broadcasting.
string	`status`	Status of the bluetooth broadcasting.

4.2.5 eon_cordova_contact object

This object allows you to access the device contacts database.

4.2.5.1 Functions

4.2.5.1.1 of_addcontactaddress

Description

Adds an address to the JSON node. It will be used when creating a new contact.

Syntax

`of_addcontactaddress` (string *`as_type`*, string *`as_formatted`*, string *`as_streetaddress`*, string *`as_locality`*, string *`as_region`*, string *`as_postalcode`*, string *`as_country`*)

Parameter

`as_type` - A string indicating what type of field this is, home for example.

`as_formatted` - The full address formatted for display. It is unsupported on iOS.

`as_streetaddress` - The full street address.

as_locality - The city or locality.

as_region - The state or region.

as_postalcode - The zip code or postal code.

as_country - The country name.

Return value

None

4.2.5.1.2 of_addcontactemail

Description

Adds an email to the JSON node. It will be used when creating a new contact.

Syntax

of_addcontactemail (string *as_email*, string *as_emailtype*, boolean *abl_emailpref*)

Parameter

as_email - Email address.

as_emailtype - Email type. Values: home, work, mobile, iPhone, or any other value that is supported by a particular device platform's contact database.

abl_emailpref - Whether used as the preference email. Values: true, false. It is unsupported on some platform.

Return value

None

4.2.5.1.3 of_addcontactname

Description

Adds a contact name to the JSON node. It will be used when creating a new contact.

Syntax

of_addcontactname (string *as_contactname*, string *as_givenname*, string *as_familyname*, string *as_middlename*, boolean *abl_pref*)

Parameter

as_contactname - The contact's complete name you want to set.

as_givenname - The contact's first name you want to set.

as_familyname - The contact's last name you want to set.

as_middlename - The contact's middle name you want to set.

abl_pref - Whether used as the preference name. Values: true, false. It is unsupported on some platform.

Return value

None

4.2.5.1.4 of_addcontactorganization

Description

Adds an organization to the JSON node. It will be used when creating a new contact.

Syntax

`of_addcontactorganization` (string *`as_name`*, string *`as_department`*, string *`as_title`*, string *`as_type`*)

`of_addcontactorganization` (string *`as_name`*, string *`as_department`*, string *`as_title`*, string *`as_type`*, boolean *`abl_pref`*)

Parameter

`as_name` - The name of the organization.

`as_department` - The department the contact works for.

`as_title` - The contact's title at the organization.

`as_type` - A string that indicates what type of field this is, home for example.

`abl_pref` - Whether used as the preference organization. Values: true, false. It is unsupported on some platform.

Return value

None

4.2.5.1.5 of_addcontactphonenumber

Description

Adds a phone number to the JSON node. It will be used when creating a new contact.

Syntax

`of_addcontactphonenumber` (string *`as_phonenumber`*, string *`as_phonetype`*, boolean *`abl_phonepref`*)

Parameter

`as_phonenumber` - Phone number.

`as_phonetype` - Phone number type. Values: home, work, mobile, iPhone, or any other value that is supported by a particular device platform's contact database.

`abl_phonepref` - Whether used as the preference number. It is unsupported on some platform.

Return value

None

4.2.5.1.6 of_addcontactphoto

Description

Adds a photo to the JSON node. It will be used when creating a new contact.

Syntax

of_addcontactphoto (string *as_photopath*, string *as_phototype*, boolean *abl_photopref*)

Parameter

as_photopath - Photo URL/path.

as_phototype - Format of the returned image: url (when the value attribute contains a URL to the photo image), or base64 (when the value contains a base64-encoded image string).

abl_photopref - Whether used as the preference photo. It is unsupported on some platform.

Return value

None

4.2.5.1.7 of_addcontacturls

Description

Adds a contact URL to the JSON node. It will be used when creating a new contact.

Syntax

of_addcontacturls (string *as_value*, string *as_type*, boolean *abl_pref*)

Parameter

as_value - The URL to be added.

as_type - The type of the URL.

abl_pref - Whether used as the preference URL. It is unsupported on some platform.

Return value

None

4.2.5.1.8 of_associatejswithpb

Description

Associates the current PowerBuilder object with a JavaScript object.

Syntax

of_associatejswithpb (string *as_contactobjectname*)

Parameter

as_contactobjectname - Name of the JavaScript object.

Return value

None

4.2.5.1.9 of_clear_jsonarray

Description

Clears the values of the JSON array.

Syntax

`of_clear_jsonarray` (eon_cjsonnodearray *`aeon_cjsonarray`*)

Parameter

`eon_cjsonnodearray` - Name of the JSON array.

Return value

None

4.2.5.1.10 of_create

Description

Creates a new contact and saves the contact to the device contacts database.

The contact information will be returned as a JSON string to the parameter of the oe_success event.

Syntax

`of_create` (string *`as_contactname`*, eon_str_cordova_contact_name *`astr_contact_name`*, eon_str_cordova_contact_address *`astr_address[]`*, eon_str_cordova_contact_organization *`astr_organization[]`*, eon_str_cordova_contact_urls *`astr_urls[]`*, eon_str_cordova_contact_field *`astr_phone[]`*, eon_str_cordova_contact_field *`astr_email[]`*, string *`as_note`*)

`of_create` (string *`as_contactname`*, string *`as_givenname`*, string *`as_familyname`*, string *`as_middlename`*, string *`as_phonenumber`*, string *`as_phonenumbertype`*, string *`as_email`*, string *`as_emailtype`*)

`of_create` (string *`as_contactname`*, string *`as_phonenumber`*)

`of_create` (string *`as_contactname`*, string *`as_phonenumber`*, string *`as_phonenumbertype`*, string *`as_email`*, string *`as_emailtype`*)

Parameter

`as_contactname` - Name of the new contact.

`astr_contact_name` - Structure for the contact name information such as first name, last name, middle name etc.

`astr_address[]` - Structure array for the contact address information such as multiple addresses.

`astr_organization[]` - Structure array for the contact organization information such as multiple organizations.

`astr_urls[]` - Structure array for the contact URL such as multiple URLs.

`astr_phone[]` - Structure array for the contact phone number such as smart phone number, home phone number etc.

`astr_email[]` - Structure array for the contact email such as multiple email addresses.

`as_note` - Note of the new contact.

`as_givenname` - First name of the new contact.

`as_familyname` - Last name of the new contact.

as_middlename - Middle name of the new contact.

as_phonenumber - Phone number of the new contact.

as_phoneNumberType - Phone number type. Values: home, work, fax, mobile.

as_email - Email address of the new contact.

as_emailType - Email type of the new contact.

Return value

String.

0: Success

error text: Error returned when failing to create the contact.

4.2.5.1.11 of_create_jsonpar

Description

Creates a contact using a JSON-format string and saves the information.

The contact information will be returned as a JSON string to the parameter of the oe_success event.

Syntax

of_create_jsonpar (string *as_contact_infor*)

Parameter

as_contact_infor - A JSON-format string that contains the contact information.

Return value

String.

0: Success

error text: Error returned when failing to create the contact.

4.2.5.1.12 of_delete

Description

Deletes the contact according to the specified name or phone number.

Syntax

of_delete (string *as_nameorphone*)

Parameter

as_nameorphone - Name or phone number of the contact to be deleted. If there are multiple contacts found, only the first one on the list will be deleted.

Return value

Integer.

1 - Success.

-1 - Failure (due to errors or other problems, or failing to find the contact).

0 - Not found.

4.2.5.1.13 of_delete_contactid

Description

Deletes the contact according to the specified contact ID.

Syntax

`of_delete_contactid` (string *`as_id`*)

Parameter

`as_id` - The unique ID of the contact to be deleted.

Return value

Integer.

1 - Success.

-1 - Failure (due to errors or other problems, or failing to find the contact).

0 - Not found.

4.2.5.1.14 of_destroy_object

Description

Destroys the JSON object.

Syntax

`of_destroy_object` ( )

Return value

None

4.2.5.1.15 of_find

Description

Finds a contact.

Syntax 1

`of_find` (string *`as_contactfield`*, string *`as_contactopt_filter`*, boolean *`abl_contactopt_multiple`*, string *`as_contactopt_desiredfields`*, boolean *`abl_contactopt_hasphonenumber`*)

Parameter 1

`as_contactfield` - The contact fields used as a search qualifier.

`as_contactopt_filter` - The search string used to find navigator.contacts.

`abl_contactopt_multiple` - Whether the find operation returns multiple navigator.contacts (default: false).

`as_contactopt_desiredfields` - The contact fields to be returned. If specified, the returned result only contains values for these fields.

abl_contactopt_hasphonenumber - Search only returns contacts with a phone number. It is supported on Android only.

Return value 1

String. Returns a JSON-format string with the found contacts.

Syntax 2

of_find (string *as_contactopt_filter*)

Parameter 2

as_contactopt_filter - The search string used to find navigator.contacts.

Return value 2

Any. eon_str_cordova_contact_find[]: if successful, the structure stores the contact name, ID, note, URL, phone number, email, address, organization, and photo; if not found, eon_str_cordova_contact_find.s_errortext = notfind; if error occurs, eon_str_cordova_contact_find.s_errortext = error.

4.2.5.1.16 of_find_contactname

Description

Finds a contact's name, ID, display name, and nick name.

Syntax

of_find_contactname (string *as_contactopt_filter*)

Parameter

as_contactopt_filter - The search string used to find navigator.contacts.

Return value

Any. eon_str_cordova_contact_find[]: if successful, the structure stores the contact name, phone number, email, address, organization, and photo; if not found, eon_str_cordova_contact_find.s_errortext = notfind; if error occurs, eon_str_cordova_contact_find.s_errortext = error.

4.2.5.1.17 of_find_event

Description

Finds a contact. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered; and the return value will be passed to the parameter of the event.

Syntax

of_find_event (string *as_contactfield*, string *as_contactopt_filter*)

of_find_event (string *as_contactfield*, string *as_contactopt_filter*, boolean *abl_contactopt_multiple*, string *as_contactopt_desiredfields*, boolean *abl_contactopt_hasphonenumber*)

of_find_event (string *as_contactfield*, string *as_contactopt_filter*, string *as_contactopt_desiredfields*)

Parameter

as_contactfield - The contact fields used as a search qualifier.

as_contactopt_filter - The search string used to find navigator.contacts.

abl_contactopt_multiple - Whether the find operation returns multiple navigator.contacts (default: false).

as_contactopt_desiredfields - The contact fields to be returned. If specified, the returned result only contains values for these fields.

abl_contactopt_hasphonenumber - Search only returns contacts with a phone number. It is supported on Android only.

Return value

None.

4.2.5.1.18 of_getcontactfieldtype

Description

Gets the contact field type. This can get all of the properties of a contact.

Syntax

of_getcontactfieldtype ( )

Return value

String.

4.2.5.1.19 of_getphoto_android

Description

Gets the photo from the device's contact database and creates a copy under the Appeon application folder.

Supported on Android only.

Syntax

of_getphoto_android (string *as_photo*)

Parameter

as_photo - The photo in the device's contact database (obtained from eon_str_cordova_contact_find[]).

Return value

String. The directory of the photo in the plugin folder of the Appeon application.

4.2.5.1.20 of_init

Description

Connects with the Cordova contacts plugin, detects if the plugin is available to call, and binds the current PowerBuilder object with the JavaScript object defined by the plugin. After that, the PowerBuilder object has all of the methods and properties that the JavaScript object has.

Syntax

`of_init ( )`

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

4.2.5.1.21 of_init_jsonobject

Description

Creates a JSON object, to help add the contact's property.

Syntax

`of_init_jsonobject ( )`

Return value

None.

4.2.5.1.22 of_remove

Description

Deletes a contact. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered; and the return value will be passed to the parameter of the event.

Syntax

`of_remove` (string *`as_contact_id`*)

Parameter

`as_contact_id` - ID of the contact that you want to remove.

Return value

None.

4.2.5.1.23 of_save

Description

Saves the newly created contact to the device. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered; and the return value will be passed to the parameter of the event.

Syntax

`of_save` (string *`as_contactobjectname`*)

Parameter

`as_contactobjectname` - Name of the contact object.

Return value

None.

4.2.5.1.24 of_setcontactaddress

Description

Creates a contactaddress JSON node and sets the value.

Syntax

`of_setcontactaddress` (string *`as_type`*, string *`as_formatted`*, string *`as_streetaddress`*, string *`as_locality`*, string *`as_region`*, string *`as_postalcode`*, string *`as_country`*)

Parameter

`as_type` - A string indicating what type of field this is, home for example.

`as_formatted` - The full address formatted for display.

`as_streetaddress` - The full street address.

`as_locality` - The city or locality.

`as_region` - The state or region.

`as_postalcode` - The zip code or postal code.

`as_country` - The country name.

Return value

eon_cjsonnode. The JSON node with the values.

4.2.5.1.25 of_setcontactfield

Description

Creates a contactfield JSON node and sets the value.

Syntax

`of_setcontactfield` (string *`as_value`*)

`of_setcontactfield` (string *`as_value`*, string *`as_type`*, boolean *`abl_pref`*)

Parameter

`as_value` - The value of the field, such as a phone number or email address.

`as_type` - In most instances, there are no pre-determined values for this parameter. For example, a phone number can specify type values of home, work, mobile, iPhone, or any other value that is supported by a particular device platform's contact database. However, for the Contact photos field, the type indicates the format of the returned image: url when the value attribute contains a URL to the photo image, or base64 when the value contains a base64-encoded image string.

`abl_pref` - Whether used as the preference value. It is unsupported on some platform.

Return value

eon_cjsonnode. The JSON node with the values.

4.2.5.1.26 of_setcontactname

Description

Sets the contact's name.

Syntax 1

`of_setcontactname` (string *`as_contactname`*)

Parameter 1

`as_contactname` - The contact's name you want to set.

Return value 1

None.

Syntax 2

`of_setcontactname` (string *`as_givenname`*, string *`as_familyname`*, string *`as_formatted`*, string *`as_middlename`*, string *`as_honorificprefix`*, string *`as_honorificsuffix`*, boolean *`abl_pref`*)

Overloads the `of_setcontactname` function for adding the other name related information.

Parameter 2

`as_givenname` - The contact's first name you want to set.

`as_familyname` - The contact's last name you want to set.

`as_formatted` - The contact's complete name you want to set.

`as_middlename` - The contact's middle name you want to set.

`as_honorificprefix` - The contact's prefix (example Mr. or Dr.)

`as_honorificsuffix` - The contact's suffix (example Esq.).

`abl_pref` - Whether used as the preference name. Values: true, false. It is unsupported on some platform.

Return value 2

eon_cjsonnode. Returns the JSON node with the values.

4.2.5.1.27 of_setcontactoptions

Description

Sets the options' value for the find parameter.

Syntax

`of_setcontactoptions` (string *`as_contactopt_filter`*, boolean *`abl_contactopt_multiple`*, string *`as_contactopt_desiredfields`*, boolean *`abl_contactopt_hasphonenumber`*)

Parameter

`as_contactopt_filter` - The string used to find navigator.contacts.

`abl_contactopt_multiple` - Whether the find operation returns multiple navigator.contacts (default: false).

`as_contactopt_desiredfields` - Contact fields to be returned.

abl_contactopt_hasphonenumber - (Android only) Search only returns contacts with a phone number (default: false).

Return value

String. Returns the JSON string with the options.

4.2.5.1.28 of_setcontactorganization

Description

Creates a contactorganization JSON node and sets the value.

Syntax

of_setcontactorganization (string *as_name*, string *as_department*, string *as_title*, string *as_type*)

Parameter

as_name - The name of the organization. It is partially supported on iOS.

as_department - The department the contact works for. It is partially supported on iOS.

as_title - The contact's title at the organization. It is partially supported on iOS.

as_type - A string that indicates what type of field this is, home for example. It is unsupported on iOS.

Return value

eon_cjsonnode. The JSON node with the values.

4.2.5.2 Properties

eon_cjsonnode iec_contactname

The JSON node object for adding the contact's name.

eon_cjsonnode iec_contactpro

The JSON node object for adding the contact's properties.

eon_cjsonnodearray iec_addressarray

The JSON array object for adding the address information.

eon_cjsonnodearray iec_emailsarray

The JSON array object for adding the email information.

eon_cjsonnodearray iec_organizationarray

The JSON array object for adding the organization information.

eon_cjsonnodearray iec_phonenumberarray

The JSON array object for adding the phone number information.

eon_cjsonnodearray iec_photoarray

The JSON array object for adding the photo information.

eon_cjsonnodearray iec_urlsarray

The JSON array object for adding the URL.

oleobject ieon_ole

PowerBuilder OLEObject object to be connected with the Cordova plugin.

powerobject ipo_bindevent

PowerBuilder object to be bound with the JavaScript object.

string is_errorevent

Stores the error event name of the PowerBuilder object.

string is_errortext

Stores the error information returned by the JavaScript function when execution failed.

string is_jserrortext

Stores the JavaScript error information when JavaScript call fails.

string is_navigatorcontacts

Stores the object name of the Contact plugin.

string is_successevent

Stores the success event name of the PowerBuilder object.

string is_successtext

Stores the value returned by the JavaScript function when execution is successful.

4.2.5.3 Structures

4.2.5.3.1 eon_str_cordova_contact_address

Description

Struct.

Stores the information of the contact address.

Property

Type	Variable Name	Description
string	`s_type`	Type of the field, home for example.
string	`s_streetaddress`	The street address.
string	`s_locality`	The city or locality.
string	`s_region`	The state or region.
string	`s_postalcode`	The zip code or postal code.
string	`s_country`	The country name.
string	`s_formatted`	The complete address.

4.2.5.3.2 eon_str_cordova_contact_field

Description

Struct.

Stores the information of the contact field.

Property

Type	Variable Name	Description
string	s_type	Type of the contact field.
string	s_value	Value of the contact field.

4.2.5.3.3 eon_str_cordova_contact_find

Description

Struct.

Stores the information of the found contact.

Property

Type	Variable Name	Description
string	s_contactname	Name of the found contact.
string	id	Unique ID of the found contact.
eon_str_cordova_contact_field	str_phonenumbers[]	Phone numbers of the found contact.
eon_str_cordova_contact_field	str_emails[]	Emails of the found contact.
string	s_photos[]	Photos of the found contact.
eon_str_cordova_contact_address	str_address[]	Address of the found contact.
eon_str_cordova_contact_organization	str_organizations[]	Organization of the found contact.
string	s_errortext	Error text.
date	d_birthday	Birthday of the found contact.
eon_str_cordova_contact_name	str_contactname	Name of the found contact.
string	s_note	Note of the found contact.
eon_str_cordova_contact_urls	str_urls[]	URLs of the found contact.

4.2.5.3.4 eon_str_cordova_contact_name

Description

Struct.

Stores the information of the contact name.

Property

Type	Variable Name	Description
string	s_givenname	First name of the contact.

Type	Variable Name	Description
string	s_formatted	Complete name of the contact.
string	s_middleName	Middle name of the contact.
string	s_familyname	Last name of the contact.
string	s_honorificPrefix	Prefix (such as Mr. or Dr.) of the contact.
string	s_honorificSuffix	Suffix (such as Esq.) of the contact.

4.2.5.3.5 eon_str_cordova_contact_organization

Description

Struct.

Stores the information of the contact organization.

Property

Type	Variable Name	Description
string	s_name	The name of the organization.
string	s_department	The department the contact works for.
string	s_title	The contact's title at the organization.
string	s_type	Type of the field, home for example.

4.2.5.3.6 eon_str_cordova_contact_urls

Description

Struct.

Stores the information of the contact URL.

Property

Type	Variable Name	Description
string	s_value	The URL.
string	s_type	The type of the URL.

4.2.6 eon_cordova_fingerprint object

On the Android-based devices, this object will open a native dialog prompting the user to authenticate using their fingerprint.

On the iOS devices, this object will scan the fingerprint of your user with the TouchID sensor.

4.2.6.1 Functions

4.2.6.1.1 of_init

Description

Connects with the Cordova fingerprint plugin, detects if the plugin is available to call, and binds the current PowerBuilder object with the JavaScript object defined by the plugin. After that, the PowerBuilder object has all of the methods and properties that the JavaScript object has.

Syntax

```
of_init()
```

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

4.2.6.1.2 of_isavailable

Description

Detects if the fingerprint/TouchID sensor works on the device.

Syntax

```
of_isavailable()
```

Return value

Boolean.

True - Fingerprint/TouchID sensor is working.

False - Fingerprint/TouchID sensor is not working.

4.2.6.1.3 of_show

Description

Opens a native dialog to use the device hardware fingerprint scanner to authenticate against fingerprints registered for the device. When the function call is successful, oe_success event will be triggered; when the function call failed, oe_error event will be triggered.

Supported on Android only.

Syntax

```
of_show()
```

Return value

None.

4.2.6.1.4 of_verifyfingerprint_ios

Description

Verifies the fingerprint function. When the function call is successful, oe_success event will be triggered; when the function call failed, oe_error event will be triggered.

Supported on iOS only.

Syntax

`of_verifyfingerprint_ios ( )`

Return value

None.

4.2.6.2 Events

4.2.6.2.1 oe_error

Description

Occurs when an error occurs during the function call. If the JavaScript function in the Cordova plugin returns any error, that error will be passed to the *`as_error`* parameter.

Syntax

`oe_error` (string *`as_error`*)

Parameter

`as_error` - A JSON-format string which contains error information returned by the function.

Return value

None

4.2.6.2.2 oe_success

Description

Occurs when the function is successfully executed. If the JavaScript function in the Cordova plugin returns any value, that value will be passed to the *`as_message`* parameter.

Syntax

`oe_success` (string *`as_message`*)

Parameter

`as_message` - A JSON-format string which contains values returned by the function. If the function called is of_isavailable, the return value indicates the fingerprint sensor is working or not.

Return value

None

4.2.6.3 Properties

boolean ib_ios

Indicates whether the platform is iOS.

oleobject ieon_ole

PowerBuilder OLEObject object to be connected with the Cordova plugin.

powerobject ipo_bindevent

PowerBuilder object to be bound with the JavaScript object.

string is_errorevent

Stores the error event name of the PowerBuilder object.

string is_errorText

Stores the error information returned by the JavaScript function when execution failed.

string is_jserrorText

Stores the JavaScript error information when JavaScript call fails.

string is_successevent

Stores the success event name of the PowerBuilder object.

string is_successText

Stores the value returned by the JavaScript function when execution is successful.

4.2.7 eon_cordova_gps object

This object provides information about the device's location, such as latitude and longitude.

4.2.7.1 Functions

4.2.7.1.1 of_configure

Description

Configures the parameters for the GPS search. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered.

Syntax

`of_configure` (eon_str_cordova_gps *`astr_gps`*)

Parameter

`astr_gps` - This structure carries the following values:

- *`desiredaccuracy`* - The desired accuracy of the geolocation system. Values: 0 (highest power, highest accuracy), 10, 100, 1000 (lowest power, lowest accuracy).
- *`stationaryRadius`* - When stopped, the minimum distance the device must move beyond the stationary location for aggressive background-tracking to engage.
- *`distanceFilter`* - The minimum distance (measured in meters) a device must move horizontally before an update event is generated.

Return value

None.

4.2.7.1.2 of_init

Description

Connects with the Cordova geolocation plugin, detects if the plugin is available to call, and binds the current PowerBuilder object with the JavaScript object defined by the plugin. After that, the PowerBuilder object has all of the methods and properties that the JavaScript object has.

Syntax

`of_init()`

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

4.2.7.1.3 of_start

Description

Enables location tracking.

Syntax

`of_start()`

Return value

None.

4.2.7.1.4 of_stop

Description

Disables location tracking.

Syntax

`of_stop()`

Return value

None.

4.2.7.2 Events

4.2.7.2.1 oe_success

Description

Occurs when the function is successfully executed. If the JavaScript function in the Cordova plugin returns any value, that value will be passed to the *as_message* parameter.

Syntax

`oe_success` (string *as_message*)

Parameter

`as_message` - A JSON-format string which contains the location data.

Return value

None

4.2.7.3 Properties

eon_str_cordova_gps_info ieon_gps_info

The structure for storing the detailed location data.

oleobject ieon_ole

PowerBuilder OLEObject object to be connected with the Cordova plugin.

powerobject ipo_bindevent

PowerBuilder object to be bound with the JavaScript object.

string is_errorevent

Stores the error event name of the PowerBuilder object.

string is_errorText

Stores the error information returned by the JavaScript function when execution failed.

string is_jserrorText

Stores the JavaScript error information when JavaScript call fails.

string is_successevent

Stores the success event name of the PowerBuilder object.

string is_successText

Stores the value returned by the JavaScript function when execution is successful.

4.2.7.4 Structures

4.2.7.4.1 eon_str_cordova_gps

Description

Struct.

Stores the information of the device GPS.

Property

Type	Variable Name	Description
long	`desiredaccuracy`	The desired accuracy of the geolocation system.
long	`stationaryradius`	When stopped, the minimum distance the device must move beyond the stationary location for aggressive background-tracking to engage.

Type	Variable Name	Description
long	distancefilter	The minimum distance (measured in meters) a device must move horizontally before an update event is generated.

4.2.7.4.2 eon_str_cordova_gps_info

Description

Struct.

Stores the information of the device's location.

Property

Type	Variable Name	Description
long	time	The timestamp of the location.
decimal { 2 }	speed	The speed.
decimal { 2 }	altitude	The altitude of the location.
decimal { 2 }	bearing	The bearing of the location
decimal { 2 }	longitude	The longitude of the location.
decimal { 2 }	latitude	The latitude of the location.
decimal { 2 }	accuracy	The accuracy of the location.
string	serviceprovider	The GPS service provider.
boolean	debug	

4.2.8 eon_cordova_orientation object

This object provides access to the device's compass. The compass is a sensor that detects the direction or heading that the device is pointed, typically from the top of the device.

4.2.8.1 Functions

4.2.8.1.1 of_clearwatch

Description

Stops watching the device's heading according to the specified watch ID.

Syntax

`of_clearwatch` (string *`as_watchid`*)

Parameter

`as_watchid` - Watch ID returned by of_watchheading.

Return value

None

4.2.8.1.2 of_getcurrentheading

Description

Gets the device's current heading/direction. When successful, the oe_success event will be triggered; when failed, the oe_error event will be triggered; and the return value will be passed to the parameter of the event.

Syntax

```
of_getcurrentheading()
```

Return value

None.

4.2.8.1.3 of_init

Description

Connects with the Cordova device orientation plugin, detects if the plugin is available to call, and binds the current PowerBuilder object with the JavaScript object defined by the plugin. After that, the PowerBuilder object has all of the methods and properties that the JavaScript object has.

Syntax

```
of_init()
```

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

4.2.8.1.4 of_watchheading

Description

Gets the device's current heading/direction at a regular interval.

Syntax

`of_watchheading` (long *`al_frequency`*, decimal *`ad_filter`*)

Parameter

`al_frequency` - How often to retrieve the device's heading in milliseconds (default: 100)

`ad_filter` - The change in degrees required to retrieve the heading. It is supported in iOS only. When this parameter is set, *`al_frequency`* is ignored.

Return value

String. Watch ID is returned and can be used by of_clearwatch.

4.2.8.2 Events

4.2.8.2.1 oe_success

Description

Occurs when the function is successfully executed. If the JavaScript function in the Cordova plugin returns any value, that value will be passed to the *as_message* parameter.

Syntax

oe_success (string *as_message*)

Parameter

as_message - A JSON-format string which contains the heading information.

Return value

None

4.2.8.3 Properties

eon_str_cordova_orientation ieon_str_orientation

The structure for storing the detailed heading information.

oleobject ieon_ole

PowerBuilder OLEObject object to be connected with the Cordova plugin.

powerobject ipo_bindevent

PowerBuilder object to be bound with the JavaScript object.

string is_errorevent

Stores the error event name of the PowerBuilder object.

string is_errorText

Stores the error information returned by the JavaScript function when execution failed.

string is_jserrorText

Stores the JavaScript error information when JavaScript call fails.

string is_successevent

Stores the success event name of the PowerBuilder object.

string is_successText

Stores the value returned by the JavaScript function when execution is successful.

4.2.8.4 Structures

4.2.8.4.1 eon_str_cordova_orientation

Description

Struct.

Stores the information of the device orientation.

Property

Type	Variable Name	Description
decimal	magneticHeading	The device's magnetic heading.

Type	Variable Name	Description
decimal	`trueHeading`	The device's true heading.
decimal	`HeadingAccuracy`	The device's heading accuracy.
decimal	`timestamp`	The device's timestamp.

4.2.9 eon_cordova_vibration object

This object provides a way to vibrate the device.

4.2.9.1 Functions

4.2.9.1.1 of_init

Description

Connects with the Cordova vibration plugin, detects if the plugin is available to call, and binds the current PowerBuilder object with the JavaScript object defined by the plugin. After that, the PowerBuilder object has all of the methods and properties that the JavaScript object has.

Syntax

`of_init` ( )

Return value

Integer.

1 - Success.

-1 - It is called in PowerBuilder or PowerServer Web, or there is an error.

4.2.9.1.2 of_vibrate

Description

Vibrates the device for a given amount of time.

Syntax

`of_vibrate` (long *`al_time`*)

`of_vibrate` (long *`al_time[]`*) (Supported on Android only)

Parameter

`al_time` - Milliseconds to vibrate the device.

`al_time[]` - Sequence of durations (in milliseconds) for which to turn on or off the vibrator.

Return value

None

4.2.9.2 Properties

oleobject ieon_ole

PowerBuilder OLEObject object to be connected with the Cordova plugin.

powerobject ipo_bindevent

PowerBuilder object to be bound with the JavaScript object.

string is_errorevent

Stores the error event name of the PowerBuilder object.

string is_errorText

Stores the error information returned by the JavaScript function when execution failed.

string is_jserrorText

Stores the JavaScript error information when JavaScript call fails.

string is_successevent

Stores the success event name of the PowerBuilder object.

string is_successText

Stores the value returned by the JavaScript function when execution is successful.

4.3 Cordova plugins (solution 2 [211])

PowerServer Mobile has packaged the following most popular Cordova plugins according to the download times on the plugin website. If you know the basics of JavaScript language and can understand the JavaScript code in the plugin, you can access these plugins using a special object Appeon creates -- **AppeonMobile.CordovaPlugin**.

Table 4.1:

Plugin Type	Name	Version	Download Page
Paypal	com.paypal.cordova.mobilesdk	3.1.23	https://www.npmjs.com/package/com.paypal.cordova.mobilesdk
Contacts	cordova-plugin-contacts	2.0.1	https://www.npmjs.com/package/cordova-plugin-contacts
Bluetooth	org.apache.cordova.bluetooth	0.9	https://github.com/tanelih/phonegap-bluetooth-plugin
Calendar	cordova-plugin-calendar	4.4.7	https://www.npmjs.com/package/cordova-plugin-calendar
File	cordova-plugin-file	4.1.1	https://www.npmjs.com/package/cordova-plugin-file
Api.ai voice-enabling engine	cordova-plugin-apiai	1.7.0	https://www.npmjs.com/package/cordova-plugin-apiai
Device	cordova-plugin-device	1.1.1	https://www.npmjs.com/package/cordova-plugin-device
Network information	cordova-plugin-network-information	1.2.0	https://www.npmjs.com/package/cordova-plugin-network-information

Plugin Type	Name	Version	Download Page
Globalization	cordova-plugin-globalization	1.0.3	https://www.npmjs.com/package/cordova-plugin-globalization
Geolocation	cordova-plugin-mauron85-background-geolocation	0.9.6	https://www.npmjs.com/package/cordova-plugin-mauron85-background-geolocation
Device orientation	cordova-plugin-device-orientation	1.0.3	https://www.npmjs.com/package/cordova-plugin-device-orientation
Media	cordova-plugin-media-capture	1.2.0	https://www.npmjs.com/package/cordova-plugin-media-capture
Vibration	cordova-plugin-vibration	2.1.0	https://www.npmjs.com/package/cordova-plugin-vibration
Battery status	cordova-plugin-battery-status	1.1.2	https://www.npmjs.com/package/cordova-plugin-battery-status
Sim	cordova-plugin-sim	1.2.1	https://www.npmjs.com/package/cordova-plugin-sim
Fingerprint authentication	Android: cordova-plugin-android-fingerprint-auth iOS: cordova-plugin-touch-id	Android: 0.0.5 iOS: 3.1.0	Android: https://www.npmjs.com/package/cordova-plugin-android-fingerprint-auth iOS: https://www.npmjs.com/package/cordova-plugin-touch-id
Bluetooth low energy	cordova-plugin-bluetoothle	4.0.0	https://www.npmjs.com/package/cordova-plugin-bluetoothle
TTS	cordova-plugin-tts	0.2.3	https://www.npmjs.com/package/cordova-plugin-tts
Clipboard	cordova.plugins.clipboard	1.1.0	https://www.npmjs.com/package/cordova-universal-clipboard
Globalization	cordova-plugin-globalization	1.0.3	https://www.npmjs.com/package/cordova-plugin-globalization
Web Socket	cordova-plugin-websocket	0.12.0	https://www.npmjs.com/package/cordova-plugin-websocket

4.3.1 Connecting with AppeonMobile.CordovaPlugin

AppeonMobile.CordovaPlugin acts as a bridge for the PowerBuilder code to execute the JavaScript code. You will need to connect with **AppeonMobile.CordovaPlugin** first and then use the functions it provides to execute the JavaScript code.

To connect/disconnect with the **AppeonMobile.CordovaPlugin** object, you can use the PowerBuilder OLEObject's `ConnectToNewObject` and `DisconnectObject` functions, for example,

```
Oleobject ole
Ole = Create Oleobject
//connects with the AppeonMobile.CordovaPlugin object
ole.Connecttonewobject ( "AppeonMobile.CordovaPlugin" )
```

```
//disconnects with the AppeonMobile.CordovaPlugin object
ole.DisconnectObject ()
```

4.3.2 AppeonMobile.CordovaPlugin object

4.3.2.1 SetTimeout function

Description

Sets the time out value in milliseconds. By default it is 5 seconds. If 0, no timeout period is in effect.

Syntax

`SetTimeout(`*`milliseconds`*`)`

Return value

None

Usage

This function is required when `ExecJavaScriptWithReturn` is called, because the Cordova plugin may connect with the network, or plugin may have bugs or invalid parameters, etc.

This function is required when @ (instead of @EventName) is used to wait for the execution result.

4.3.2.2 AssociateJSwithPB function

Description

Associates a Cordova plugin (JavaScript object) with a PowerBuilder object. After associated, the PowerBuilder object will have all of the methods and properties that the JavaScript object has.

Syntax

`AssociateJSwithPB(`*`string`*`,`*`PBobject`*`)`

Parameter

`string` - The JavaScript object defined by the Cordova plugin. If it is not a global object, then it will need to be created first before it can be associated with.

`PBobject` - The PowerBuilder object. If callback functions are used in the JavaScript object, then the corresponding callback events should be defined in this object.

Return value

None

Code example

```
Oleobject ole
Ole = Create Oleobject
//Connects with AppeonMobile.CordovaPlugin
ole.Connecttonewobject ( "AppeonMobile.CordovaPlugin" )
//Associates with the plugin object
//navigator.contacts is a global object according to the Cordova help
```

```
ole.AssociateJSwithPB ( "navigator.contacts", this )
```

4.3.2.3 ExecJavaScript function

Description

Executes a line of JavaScript code and returns no value. This function can be used to execute a line of JavaScript code that returns no value (asynchronous execution).

To execute a line of JavaScript code that returns a value, call the `ExecJavaScriptWithReturn` function.

To execute a line of JavaScript code that contains a callback function, it is not recommended to use `ExecJavaScript`, as it will be very complicated to write scripts to use the callback function in `ExecJavaScript`; instead, it is recommended that the user associates with the JavaScript object where the function will be called, and then use dot notation to execute this function. Dot notation will be described more in Using dot notation.

Syntax

`ExecJavaScript(`*`string`*`)`

Return value

None

Usage

This function must be used to execute the JavaScript code that creates an object. If the JavaScript code contains a string, then the string must be included in double quotation marks. The semi colon (;) at the end of the JavaScript code can be preserved or omitted.

Example 1:

JavaScript code:

```
var myContact = navigator.contacts.create ({"displayName": "Test User"});
```

Corresponding PowerBuilder code:

```
oleobject.ExecJavaScript ('var mycontact = navigator.contacts.create
 ({"displyaName":"TestUser"});')
```

Example 2:

JavaScript code:

```
var options = new ContactFindOptions ();
```

Corresponding PowerBuilder code:

```
lole_cordova.ExecJavaScript ("var options = new ContactFindOptions();")
```

Code example

```
Oleobject lole_cordova
lole_cordova = Create OleObject
//connects with AppeonMobile.CordovaPlugin
lole_cordova.connecttonewobject ( "AppeonMobile.CordovaPlugin" )
//associates this PB object with a JS object navigator.contacts
lole_cordova.AssociateJSwithPB ( "navigator.contacts", this )
//sets the timeout value
```

```
lole_cordova.SetTimeout (10000)
//executes a line of JS code
//creates a new ContactFindOptions object
lole_cordova.ExecJavaScript ( "var options = new ContactFindOptions();" )
//associates with the new object
lole_cordova.AssociateJSwithPB ( "options", This )
//as the value of filter is a string, include "Test" in quotation marks
lole_cordova.ExecJavaScript ( "options.filter=~"Test~"" )
//as the multiple object is a boolean, no need to include it in quotation marks
lole_cordova.ExecJavaScript ( "options.multiple=true" )
```

4.3.2.4 ExecJavaScriptWithReturn function

Description

Executes a line of JavaScript code and returns the result.

Syntax

`ExecJavaScriptWithReturn(` *`string`* `)`

Return value

String.

Returns a string if successful. If an object is returned, the object properties will be described in a JSON-format string.

Returns an empty string if timeout is reached (possible causes for timeout: script is too long, script has errors, parameter is incorrect, etc.).

Code example

If the JavaScript code contains a string, then the string must be included in double quotation marks. The semi colon (;) at the end of the JavaScript code can be preserved or omitted.

```
String ls_Return
Oleobject lole_cordova
lole_cordova = Create OleObject
//connects the PB OLEObject with AppeonMobile.cordovaPlugin
lole_cordova.connecttonewobject ("AppeonMobile.CordovaPlugin")
//associates this PB object with a JS object navigator.contacts
lole_cordova.AssociateJSwithPB ("navigator.contacts", this)
//sets the timeout value
lole_cordova.SetTimeout (10000)
//executes a line of JS code and returns no value
lole_cordova.ExecJavaScript ("var options = new ContactFindOptions();")
//executes a line of JS code and returns a value. The value is a string in the JSON
 format containing all of the properties of options object
ls_Return = lole_cordova.ExecJavaScriptWithReturn ("options")
```

4.3.2.5 ExecJavaScriptGetLastFuncCallRetVal function

Description

Gets the return value of the last method that uses callback functions. Executing this function will get the return value of the last executed JavaScript method that uses callback functions.

Syntax

`ExecJavaScriptGetLastFuncCallRetVal()`

Return value

String.

Code example

```
oleobject ole_orientation
ole_orientation = Create oleobject
Integer li_Return
String ls_Return
li_Return = ole_orientation.connecttonewobject ("AppeonMobile.CordovaPlugin")
ole_orientation.AssociateJSwithPB ("navigator.compass", this)
//ls_Return stores the return value of the callback event
ls_Return = ole_orientation.watchHeading ('@', '@', '{frequency: 3000}')
//gets the return value of watchHeading()
ls_Return = ole_orientation.ExecJavaScriptGetLastFuncCallRetVal()
ole_orientation.watchHeading ('@ue_contactfind', '@ue_contactfind_error',
 '{frequency: 50}')
//gets the return value of watchHeading()
ls_Return = ole_orientation.ExecJavaScriptGetLastFuncCallRetVal()
ole_orientation.disconnectobject()
```

4.3.2.6 GetLastErrorExecJS function

Description

Gets the error information if the JavaScript code has an error, including incorrect function name, misspellings, execution errors etc. The error information is returned from the JavaScript code. You can use `GetLastErrorExecJS` together with `oe_execjserror` (string *`errormessage`*) to help determine the cause.

Syntax

`GetLastErrorExecJS()`

Return value

String.

The following error types can be detected by `GetLastErrorExecJS`:

- Missing "@" when using the callback event
- Misspelling of the JavaScript function name or object property name in a dot notation
- Misspelling of the Appeon-provided function name
- Incorrect JavaScript object name or function name such as case mismatch
- Format error of the JSON string, such as missing open bracket ("[")
- Incorrect parameter or invalid property value in a dot notation of calling JavaScript code

The following error types **cannot** be detected by `GetLastErrorExecJS`:

- Missing "@" if you do not want to use the callback event
- Misspelling of the JavaScript object property name
- Incorrect keyword name in a JSON string

4.3.2.7 oe_execjserror event

Description

Occurs when the JavaScript code has an error, including incorrect function name, misspellings, and execution errors etc. It is an asynchronous event.

Syntax

`oe_execjserror` (string errormessage)

Usage

To use this event, you need to add an event with the same name in the PowerBuilder object that is bound with the JavaScript object via `AssociateJSwithPB` (*`string`*, *`PBobject`*).

4.3.3 Accessing the JavaScript object properties/methods

4.3.3.1 Using dot notation

After associating a PowerBuilder object with a JavaScript object, you can access the JavaScript object properties and methods using dot notation (.).

Regarding the return value, only when the method uses a callback function, the method's return value will be returned as a string. If the method uses no callback function, no value will be returned; if you want to get the return value, you can execute the JavaScript code.

To access a JavaScript method:

JavaScript code:

```
navigator.contacts.find (fields, onSuccess, onError);
```

The corresponding PowerBuilder code:

```
//creates and connects the PB OLEObject with AppeonMobile.cordovaPlugin. Scripts
 are omitted here.
//associates this PB object with a JS object
oleobject.AssociateJSwithPB ("navigator.contacts", this)
//executes the find method. The method parameter will be described in detail later.
//ls_Return is a JSON-format string. The string contains the information of the
 found contacts including name, phone number, address etc.
ls_Return = oleobject.find (fields, "@", "@")
```

To access a JavaScript property:

JavaScript code:

```
//create the object
var optionsreturn = new ContactFindOptions();
//sets the property value
optionsreturn.filter = "Bob"; //property value is a string
//obtains the property value
var filtervalue = optionsreturn.filter;
```

The corresponding PowerBuilder code:

```
String filtervalue
//creates the object
oleobject.ExecJavaScript ("var optionsreturn = new ContactFindOptions();")
```

```
//associates the object
oleobject.AssociateJSwithPB ("optionsreturn", this)
//accesses the property
//sets the property value
oleobject.filter ="~"Bob~"" //string data type, included in quotation marks
//obtains the property value
filtervalue = oleobject.filter
```

4.3.3.2 Using a JSON string

You can use a JSON string as the parameter to set the object property. You will first need to know the JSON format and the data type of the property values. For example,

Suppose there is an "options" object parameter in the JavaScript method:

```
navigator.contacts.find (fields, onSuccess, onError, options);
```

You can set the value for the "options" object parameter using a JSON string in PowerBuilder code like this:

```
ls_Return = oleobject.find (fields, "@", "@", '{"filter":"Bob"}')
```

Here is another example:

JavaScript code:

```
var options = new ContactFindOptions();
options.filter = "Bob";
```

The corresponding PowerBuilder code (sets the object property using a JSON string when creating the object):

```
oleobject.ExecJavaScript('var options = new ContactFindOptions({"filter":"Bob"});')
```

4.3.3.3 Using ExecJavaScript/ExecJavaScriptWithReturn function

You can call the `ExecJavaScript`/`ExecJavaScriptWithReturn` function of **AppeonMobile.CordovaPlugin** object to execute the JavaScript code that sets the object properties, for example,

```
//creates the object
ls_Return = oleobject.ExecJavaScript ("var optionsreturn = new
 ContactFindOptions();")
//sets the value of the object property
ls_Return = oleobject.ExecJavaScriptWithReturn ("optionsreturn.filter=~"test1~"")
//obtains the property value and stores in a variable
oleobject.ExecJavaScript("var testfilter = optionsreturn.filter")
//gets the variable value and returns the value
ls_Return = oleobject.ExecJavaScriptWithReturn ("testfilter")
```

4.3.3.4 Using eon_cjsonnode/eon_cjsonnodearray object

eon_cjsonnode and **eon_cjsonnodearray** objects are provided in the Appeon Workarounds PBL for handling the JSON-format strings. These two objects can be used to access the JS object properties if the object property structure are complex, as it may be very difficult to get the value from the returned string which is long and complex. You can determine which object to use according to the brace/bracket used in the string:

- values in braces ({}), node type, use **eon_cjsonnode**

- values in brackets ([]), array type, use **eon_cjsonnodearray**

Here is an example that shows how to obtain all of the phone numbers from the following JSON string:

```
JSONstring = {"displayName":"test1", "phoneNumbers":[{"type":"work",
 "value":"212-555-1234", "pref":"false"}, {"type":"mobile", "value":"917-555-5432",
 "true":"false"}, {"type":"home", "value":"203-555-7890", "pref":"false"}]}
```

Use **eon_cjsonnode**, as the string is in a node (in {}):

```
Integer li_count //store the total amount of phone numbers
Eon_cjsonnode returnnode //store the returned string
returnnode = create Eon_cjsonnode // create the JSON node object
//formatize the string in the JSON object
returnnode.of_load (JSONstring)
//phoneNumbers is an array, so use Eon_cjsonnodearray to receive values
//You can also use of_gettypebykey to determine the node type.
Eon_cjsonnodearray phoneNumberarray //phoneNumbers is the node value, containing
 multiple phone numbers
phoneNumberarray = create Eon_cjsonnodearray
phoneNumberarray = returnnode.of_ valuearray ("phoneNumbers")
li_count = phoneNumberarray.of_size()
Eon_cjsonnode phoneNumber //store the returned string
phoneNumber = create Eon_cjsonnode
For li_for = 1 to li_Count
 phoneNumber = phoneNumberarray.Of_getnodebyinex (li_For)
 messagebox ('phonenumber', phoneNumber.of_valuestring (value)) //export the
 obtained phone numbers
Next
```

4.3.3.5 Code example

Here is a step-by-step guide with code examples that shows how to use different ways to set a JavaScript property.

Step 1: Create the PowerBuilder **OLEObject** object, connect with the **AppeonMobile.CordovaPlugin** object, and associate the PowerBuilder object with the JavaScript object (**navigator.contacts**).

```
Oleobject lole_cordova
lole_cordova = Create OleObject
li_Return = lole_cordova.connecttonewobject ("AppeonMobile.CordovaPlugin")
lole_cordova.AssociateJSwithPB ("navigator.contacts", this)
```

Step 2: Add the properties and values for the **options** object.

Using a JSON string:

1. Create the **iec_option** object from the **eon_cjsonnode** object.

   ```
   eon_cjsonnode iec_option
   iec_option = Create eon_cjsonnode
   ```

2. Call the **iec_option**'s function to add the property and value for the **options** object.

   ```
   iec_option.of_addkeyvalue ("filter","test")
   iec_option.of_addkeyvalue ("multiple",true)
   iec_option.of_addkeyvalue ("hasPhoneNumber",false)
   ```

3. Call the **iec_option**'s function to return the JSON string.

```
ls_Return = iole_contact.find (["name"], "@", "@", iec_option.of_ToString() )
```

Using dot notation:

1. Create the **options** object.

```
lole_cordova.ExecJavaScript ("var options = new ContactFindOptions();")
ls_Return = lole_cordova.ExecJavaScriptWithReturn ("options")
```

2. Associate the PowerBuilder object with the **options** object.

```
lole_cordova.AssociateJSwithPB ("options", This)
```

3. Set the values of the filter, multiple and hasPhoneNumber properties for the **options** object.

```
//string must be in double quotes
lole_cordova.filter = "~"test~""  //'"test"' is also valid
lole_cordova.multiple = true
lole_cordova.hasPhoneNumber = false
ls_Return = lole_cordova.ExecJavaScriptWithReturn ("options")
```

4. Associate the PowerBuilder object with the **navigator.contacts** object again.

```
lole_cordova.AssociateJSwithPB ("navigator.contacts", this)
ls_Return = lole_cordova.find (sle_1.Text, "@", "@", "options" )
```

Using ExecJavaScript/ExecJavaScriptWithReturn function:

1. Create the **options** object.

```
lole_cordova.ExecJavaScript ("var options = new ContactFindOptions();")
ls_Return = lole_cordova.ExecJavaScriptWithReturn ("options")
```

2. Associate the PowerBuilder object with the **options** object.

```
lole_cordova.AssociateJSwithPB ("options", This)
```

3. Set the values of the filter, multiple and hasPhoneNumber properties for the **options** object.

```
lole_cordova.ExecJavaScript ("options.filter=~"tes~t"")
lole_cordova.ExecJavaScript ("options.multiple=true")
lole_cordova.ExecJavaScript ("options.hasPhoneNumber=true")
ls_Return = lole_cordova.ExecJavaScriptWithReturn ("options")
```

4. Associate the PowerBuilder object with the **navigator.contacts** object again.

```
lole_cordova.AssociateJSwithPB ("navigator.contacts", this)
ls_Return = lole_cordova.find (sle_1.Text, "@", "@", "options")
```

4.3.4 Accessing the JavaScript array

Text strings must be included in double quotation marks, others (object names, boolean values, numeric values etc.) need not. This also complies with the JSON format.

Here is an example that uses the `contacts.create` method to add phoneNumbers. According to the Cordova help, phoneNumbers is the property of contact and its data type is an array ContactField[] with three properties: type, value, & pref.

JavaScript code:

```
//creates a new contact
var contact = navigator.contacts.create();
//stores the contact phone numbers in ContactField[]
var phoneNumbers = [];
phoneNumbers[0] = new ContactField ('work', '212-555-1234', false);
phoneNumbers[1] = new ContactField ('mobile', '917-555-5432', true); // preferred
 number
phoneNumbers[2] = new ContactField ('home', '203-555-7890', false);
contact.phoneNumbers = phoneNumbers;
// save the contact
contact.save();
```

The corresponding PowerBuilder code (using dot notation):

```
oleobject.ExecJavaScript ( "var contact = navigator.contacts.create();" )
oleobject.AssociateJSwithPB ( 'contact', This )
//creates the ContactField object
oleobject.ExecJavaScript ( "var phoneNumber1 = new ContactField();" )
//associates with the object
oleobject.AssociateJSwithPB ( "phoneNumber1", this );
//sets the property values
oleobject.value = '"212-555-1234"'
oleobject.type = '"work"'
oleobject.pref = 'false'

//creates the ContactField object
oleobject.ExecJavaScript ( "var phoneNumber2 = new ContactField();" )
//associates with the object
oleobject.AssociateJSwithPB ( "phoneNumber2", this );
//sets the property values
oleobject.value = '"917-555-5432"'
oleobject.type = '"mobile"'
oleobject.pref = 'true'

//creates the ContactField object
oleobject.ExecJavaScript ( "var phoneNumber3 = new ContactField();" )
//associates with the object
oleobject.AssociateJSwithPB ( "phoneNumber3", this );
//sets the property values
oleobject.type = '"home"'
oleobject.value = '"203-555-7890"'
oleobject.pref = 'false'

//adds phoneNumbers to the newly created contact object
oleobject.AssociateJSwithPB ( "contact", this ) //associates with the contact
oleobject.phoneNumbers = '[phoneNumber1, phoneNumber2, phoneNumber3]'
oleobject.save ( "@", "@" )
```

The corresponding PowerBuilder code (using JSON):

```
oleobject.ExecJavaScript ( 'var contact = navigator.contacts.create
 ({"phoneNumbers":[{"type":"work","value":"212-555-1234","pref":"false"},
 {"type":"mobile","value":"917-555-5432","true":"false"},
 {"type":"home","value":"203-555-7890","pref":"false"}]});' )
```

4.3.5 Using callback functions

Callback function is very common in JavaScript; however there is no callback function in PowerScript. And in order to support the JavaScript callback function, a corresponding PowerScript event must be created and be called back instead (let's call it callback event). Which means, for each JavaScript callback function, define an event in the associated

PowerBuilder object, and use **@EventName** to replace the JavaScript callback function name. If you do not want to use the callback event in PowerBuilder, use **@**, then the function will return the result directly when the execution is complete. The return value is a string (if an object is to be returned, then the object properties will be returned in a JSON string).

- **EventName** is the name of the event of the object that the second parameter of the `AssociateJSwithPB` function points to.
- The event will need to have a parameter for receiving the value returned by the function.
- If the function uses more than one callback event, either all of them are used, or none; it cannot be some used, while others not, because the event to be called back will be in asynchronous execution. For example, the following script is incorrect:

```
iole_contact.find ( '["name"]', "@ue_contactfind", "@",
 iec_option.of_ToString() )
```

- When the callback event failed to execute, an error code is returned. The error code is defined in the Cordova help, and you will need to search the error code in the Cordova help to determine the cause.
- As the callback event is executed asynchronously, do not execute `OLEObject.DisconnectObject()` right after **@+EventName** is used, otherwise the callback event may fail.

Here is a code example.

The JavaScript callback function `onSuccess`:

```
function onSuccess(contacts) {
    for (var i = 0; i < contacts.length; i++) {
        for (var j = 0; j < contacts[i].addresses.length; j++) {
            alert("Pref: "         + contacts[i].addresses[j].pref          + "\n"
 +
                "Type: "           + contacts[i].addresses[j].type          + "\n"
 +
                "Formatted: "      + contacts[i].addresses[j].formatted     + "\n"
 +
                "Street Address: " + contacts[i].addresses[j].streetAddress + "\n"
 +
               "Locality: "       + contacts[i].addresses[j].locality      + "\n" +
                "Region: "         + contacts[i].addresses[j].region        + "\n"
 +
                "Postal Code: "    + contacts[i].addresses[j].postalCode    + "\n"
 +
                "Country: "        + contacts[i].addresses[j].country);
        }
    }
};
```

The JavaScript callback function `onError`:

```
function onError(contactError) {
    alert('onError!');
};
```

The JavaScript method that uses these two callback functions:

```
//finds all contacts
```

```
var options = new ContactFindOptions();
options.filter = "";
options.multiple = true;
var filter = ["displayName", "addresses"];
navigator.contacts.find (filter, onSuccess, onError, options);
```

The PowerBuilder callback event `ue_contactfind` (the counterpart of `onSuccess`) defined in the associated PowerBuilder object:

```
ue_contactfind (string as_return) return none
```

The PowerBuilder callback event `ue_contactfind_error` (the counterpart of `onError`) defined in the associated PowerBuilder object:

```
ue_contactfind_error (string as_return) return none
```

The PowerBuilder code that uses these two callback events:

```
Oleobject lole_cordova
lole_cordova = Create OleObject
li_Return = lole_cordova.connecttonewobject ( "AppeonMobile.CordovaPlugin" )
//associates with the current PowerBuilder object
lole_cordova.AssociateJSwithPB ("navigator.contacts", this)
//ue_contactfind & ue_contactfind_error defined in "this" object (the currently
 associated PB object)
//iec_option.of_ToString() is a JSON string
ls_Return = iole_contact.find ('["name"]', "@ue_contactfind",
 "@ue_contactfind_error", iec_option.of_ToString() )
//or replace it with the following script, if not to use any callback event
//ls_Return = iole_contact.find ('["name"]',"@", "@", iec_option.of_ToString() )
```

4.3.6 Code examples

Let's take Contact Cordova plugin as an example to show you how to add a contact to the mobile device and how to write the PowerBuilder code.

The corresponding Cordova help can be found here: https://www.npmjs.com/package/cordova-plugin-contacts.

Example 1: To add the following contact to the device:

name: Tester

phone number: 15845562007, 07552698563

Step 1: Create a PowerBuilder **OLEObject** object and call the `ConnectToNewObject` function to connect it to the **AppeonMobile.CordovaPlugin** object.

```
Oleobject ole_cordova
ole_cordova = Create OLEObject
ole_cordova.ConnectToNewObject ("AppeonMobile.CordovaPlugin")
```

Step 2: Call the **AppeonMobile.CordovaPlugin**'s `AssociateJSwithPB` function to associate the JavaScript object with the current PowerBuilder object.

navigator.contacts is the name of the JavaScript object defined in the Contact Cordova plugin. You can get this name from the Cordova help.

this represents the object where the callback function used by **navigator.contacts** should be transferred to.

```
ole_cordova.AssociateJSwithPB ("navigator.contacts", this)
```

Step 3: Create a new Contact object -- **mycontact**.

Cordova help has this: "The navigator.contacts.create method is synchronous, and returns a new Contact object."

```
ole_cordova.ExecJavaScript ('var mycontact = navigator.contacts.create();')
```

Step 4: Set the name of the new contact person to "Tester".

According to the Cordova help, the **Contact** object has three properties for name: displayName, name, & nickname. We choose to use nickname to set the new contact's name. These three properties are supported differently by different platform. You can find this out in the Cordova help and determine which property to use according to your request.

```
ole_cordova.ExecJavaScript ('mycontact.nickname = "Tester";')
```

Step 5: Set the phone number of the new contact person to "15845562007" & "07552698563".

According to the Cordova help, the **Contact** object has **ContactField[]** (an array) for phoneNumbers.

Create the **ContactField** object first. According to Cordova help, the **ContactField** object has three properties: type, value, & pref. Then set the phone number to 15845562007:

```
ole_cordova.ExecJavaScript ("var p1 = new ContactField();") //creates the object
ole_cordova.ExecJavaScript ("p1.type = 'home';") //value is a string, so include it
 in quotes
ole_cordova.ExecJavaScript ("p1.value = '15845562007';")
ole_cordova.ExecJavaScript ("p1.pref = true;")
```

And set the phone number to 07552698563:

```
ole_cordova.ExecJavaScript ("var p2 = new ContactField();")
ole_cordova.ExecJavaScript ("p2.type = 'work';")
ole_cordova.ExecJavaScript ("p2.value = '07552698563';")
ole_cordova.ExecJavaScript ("p2.pref = false;")
```

Finally, set the values in an array for the phoneNumbers property:

```
ole_cordova.ExecJavaScript ("mycontact.phoneNumbers = [p1,p2];")
```

Step 6: Save the new **Contact** object and the values.

According to the Cordova help, after the new **Contact** object is created, it must be saved using the `save()` method in order for the values to be saved into the database.

The `save()` method uses callback functions, so it cannot be executed using `ExecJavaScript`. We need to associate with the **mycontact** object to execute the `save()` method.

```
ole_cordova.AssociateJSwithPB ("mycontact", this)

String ls_Return //for receiving the returned value from save().
//The new contact information that is successfully saved will be returned in a JSON
 string.
ls_Return = ole_cordova.save ("@","@") //we choose not to use callback function
```

Example 2: to write the PowerBuilder code according to the JavaScript code.

The following table shows you how to write the PowerBuilder code that will execute and get the same result as the JavaScript code.

The JavaScript code uses **ContactField** to set phoneNumbers, and it is copied from the Cordova help: https://www.npmjs.com/package/cordova-plugin-contacts.

Table 4.2: JavaScript-to-PowerScript mapping

JavaScript	PowerScript

None	The following preparation is required in the PowerBuilder only.
	Create an **OLEObject** object, connect it to **AppeonMobile.CordovaPlugin**, and associate the PB object with the JS object.
	`Oleobject ole_cordova`
	`ole_cordova = Create OLEObject`
	`ole_cordova.ConnectToNewObject ("AppeonMobile.CordovaPlugin")`
	`ole_cordova.AssociateJSwithPB ("navigator.contacts", this)`

Create a new contact: `var contact = navigator.contacts.create();`	Use `ExecJavaScript` to directly execute the JavaScript code: `ole_cordova.ExecJavaScript ('var contact = navigator.contacts.create();')`

Store contact phone numbers in **ContactField[]**:
`var phoneNumbers = [];`
`phoneNumbers[0] = new ContactField ('work', '212-555-1234', false);`
`phoneNumbers[1] = new ContactField ('mobile', '917-555-5432', true);`
`phoneNumbers[2] = new ContactField ('home', '203-555-7890', false);`
`contact.phoneNumbers = phoneNumbers;`
	Use `ExecJavaScript` to directly execute the JavaScript code:
	`ole_cordova.ExecJavaScript ('var phoneNumbers = [];')`
	`ole_cordova.ExecJavaScript ('phoneNumbers[0] = new ContactField ("work", "212-555-1234", false);')`
	`ole_cordova.ExecJavaScript ('phoneNumbers[1] = new ContactField ("mobile", "917-555-5432", true);')`
	`ole_cordova.ExecJavaScript ('phoneNumbers[2] = new ContactField ("home", "203-555-7890", false);')`
	`ole_cordova.ExecJavaScript ('contact.phoneNumbers = phoneNumbers;')`

Save the contact:
`contact.save();`
	Associate the JavaScript **Contact** object with the PB **OLEObject** object, and then use dot notation to execute `save()`.
	`save()` uses callback functions, so it cannot be executed by `ExecJavaScript`.
	`ole_cordova.AssociateJSwithPB ("contact", this)`
	`String ls_Return //for receiving the save() returned value`
	`//the new contact information will be returned in a JSON string`
	`ls_Return  = ole_cordova.save ("@", "@")`

5 PowerServer open interfaces

5.1 PowerServer open interfaces

5.1.1 Overview

PowerServer open interfaces give users the opportunity to manage the client, user sessions and transactions for their Appeon-deployed applications. PowerServer open interfaces are encapsulated as the server-side component and Web service in the PowerServer, therefore, you can call them in the following different ways:

- Method 1: Call the interface through the non-autoinstantiated NVO provided by Appeon Workarounds PBL. **This method allows you to call the interface in the Appeon-deployed application only.**

  - To call the open interfaces in PowerServer installed to the .NET IIS server, you will need to call the interface through the .NET component (AppeonDotNetComponent).

  - To call the open interfaces in PowerServer installed to the Java application server (such as JBoss, WebLogic, WebSphere, or JEUS), you will need to call the interface through the EJB component (EJBObject).

  To use the .NET or EJB component provided by Appeon and make the code effective, you will need to

  1. add the Appeon Workarounds PBLs to your PowerBuilder application; and then

  2. deploy your PowerBuilder application to PowerServer the same way you would deploy a normal PowerBuilder application.

- Method 2: Call the open interface Web service provided by PowerServer. **This method allows you to call the interface from applications written in PowerScript, Java, or C# (but not Appeon-deployed application).**

  The Web service provided by PowerServer is a standard Web service, therefore, you can call the interface in the PowerBuilder application, Java application, C# application etc. Authentication is required when calling this Web service; the user name and password is the same as that used for accessing the AEM.

  Only the following interfaces (not all of the PowerServer open interfaces) are provided as Web service: getAllClients, getAllSessions, getSessionbyID, getSessionCount, and getServerVersion. The interface name is case-sensitive when called by the other programming language rather than PowerScript.

  To call and make the interface effective in the PowerBuilder application, you will need to generate the client proxy object via the .NET Web service engine.

  - URL for the open interface Web service in PowerServer installed to the Java application server (such as JBoss, WebLogic, WebSphere, or JEUS): http://*AppeonServerName*:*Port*/servlet/AppeonService?wsdl

- URL for the open interface Web service in PowerServer installed to the .NET IIS server: http://*AppeonServerName*/servlet/AppeonService.asmx

5.1.2 getAllClients

Description

getAllClients gets the IP addresses of all client machines which corresponds to the active sessions that are opened for the specified application in the specified PowerServer.

5.1.2.1 Syntax for .NET component

proxyobject.of_execinterface ("**getAllClients**", ref any *paralist[]*)

Table 5.1:

Argumen	Description
proxyobject	The instance of the proxy object generated for AppeonDotNetComponent.
paralist[]	Arrays of Any type. Specifies the parameter arrays for the component function. *paralist[1]* - The name of the PowerServer in which the active sessions are created. The value should be IP address or IP address with port number, for example, 192.0.0.123, or 192.0.0.123:8080, and separated with semicolons (;) if there are multiple servers (for example in a PowerServer cluster). If *paralist[1]* is NULL or an empty string (""), **getAllClients** returns the IP address of all clients for the specified application in all servers in a PowerServer cluster (you will need to configure the cluster in AEM first). *paralist[2]* - The name of the application that is deployed to the specified PowerServer. If *paralist[2]* is NULL or empty string (""), **getAllClients** returns the IP address of clients for all applications in the specified PowerServer. *paralist[3]* - The return value of **getAllClients** function: the IP address of all clients. This parameter must be declared, but need not to be assigned with values, as it is used to hold the return value.

Return value of of_execinterface

Returns 1 if the **of_execinterface** function succeeds and one of the following negative values if an error occurs.

-1: function fails to execute

-2: the specified PowerServer is not found

-3: the specified application is not found

Return value of getAllClients

Returns the IP address if the **getAllClients** function succeeds and returns an empty string if an error occurs.

The return value is stored in the *paralist[3]* parameter of the **of_execinterface** function.

Code example

```
Any la_l[]
```

```
Long lRet
...
la_l[1] = ls_servername
la_l[2] = ls_appname
la_l[3] = ls_clientcontent

lRet = loadDBList.of_ExecInterface ("getAllClients", la_l) //result is returned to
 la_l[3]
```

5.1.2.2 Syntax for EJB component

io_ejb.regstring (String *serverName*)

io_ejb.regstring (String *appName*)

io_ejb.Invokeretstring (long *objid*, "**getAllClients**", true, ref string *retval*)

Table 5.2:

Argumen	Description
io_ejb	A reference of Appeon EJBobject.
serverNam	The name of the PowerServer in which the active sessions are created. The value should be IP address or IP address with port number, for example, 192.0.0.123, or 192.0.0.123:8080, and separated with semicolons (;) if there are multiple servers (for example in a PowerServer cluster). If *serverName* is NULL or an empty string (""), **getAllClients** returns the IP address of all clients for the specified application in all servers in an PowerServer cluster (you will need to configure the cluster in AEM first).
appNam	The name of the application that is deployed to the specified PowerServer. If *appName* is NULL or empty string (""), **getAllClients** returns the IP address of clients for all applications in the specified PowerServer.
objid	The handle to the component method.
retval	The return value of **getAllClients** function: the IP address of all clients. This parameter must be declared, but need not to be assigned with values, as it is used to hold the return value.

Return value of Invokeretstring

Returns 1 if the **Invokeretstring** function succeeds and one of the following negative values if an error occurs.

-1: function fails to execute

-2: the specified PowerServer is not found

-3: the specified application is not found

Return value of getAllClients

Returns the IP address if the **getAllClients** function succeeds and returns an empty string if an error occurs.

The return value is stored in the *retval* parameter of **Invokeretstring** function.

Code example

Note that **invokeretstring** function is called because the return value is a string.

```
string retval01
string ls_msg
...
io_ejb.regstring (ls_servername)
io_ejb.regstring (ls_appname)

ls_msg = io_ejb.invokeretstring (il_bean1, "getAllClients", true, ref retval01) //
result is returned to retval01
```

5.1.2.3 Syntax for Web Service

getAllClients (string *serverName*, string *appName*)

Table 5.3:

Argumen	Description
serverNam	The name of the PowerServer in which the active sessions are created. The value should be IP address or IP address with port number, for example, 192.0.0.123, or 192.0.0.123:8080, and separated with semicolons (;) if there are multiple servers (for example in a PowerServer cluster). If *serverName* is NULL or an empty string (""), **getAllClients** returns the IP address of all clients for the specified application in all servers in an PowerServer cluster (you will need to configure the cluster in AEM first).
appNam	The name of the application that is deployed to the specified PowerServer. If *appName* is NULL or empty string (""), **getAllClients** returns the IP address of clients for all applications in the specified PowerServer.

Return value

Returns the IP address if the function succeeds and returns an empty string if an error occurs.

Code example

```
string ls_return,ls_server,ls_app
ieon_soap.createinstance (ieon_ws, 'wsappeonservice')
ieon_head.username = 'admin'
ieon_head.password = 'admin'
ieon_ws.setauthsoapheadervalue (ieon_head)
ls_return = ieon_ws.getAllClients (ls_server, ls_app)
```

5.1.3 getAllSessions

Description

getAllSessions returns the detail information of active sessions with XML format, which are opened for the specific application in the specific PowerServer.

Syntax

5.1.3.1 Syntax for .NET component

proxyobject.of_execinterface ("**getAllSessions**", ref any *paralist[]*)

Table 5.4:

Argumen	Description
proxyobject	The instance of the proxy object generated for AppeonDotNetComponent.

Argumen	Description
paralist[]	Arrays of Any type. Specifies the parameter arrays for the component function. *paralist[1]* - The name of the PowerServer that the sessions are created in. The value should be IP address or IP address with port number, for example, 192.0.0.123, or 192.0.0.123:8080, and separated with semicolons (;) if there are multiple servers (for example in a PowerServer cluster). If *paralist[1]* is NULL or an empty string (""), **getAllSessions** returns the information of all active sessions in a PowerServer cluster (you will need to configure the cluster in AEM first). *paralist[2]* - The name of the application that is deployed to the PowerServer. If *paralist[2]* is NULL or an empty string (""), **getAllSessions** returns the information of active sessions opened for all applications in the specified PowerServer. *paralist[3]* - The return value of **getAllSessions** function: the session content in XML format. This parameter must be cleared, but need not to be assigned with values, as it is used to hold the return value.

Return value of of_execinterface

Returns 1 if the **of_execinterface** function succeeds and one of the following negative values if an error occurs.

-1: function fails to execute

-2: the specified PowerServer is not found

-3: the specified application is not found

Return value of getAllSessions

Returns an XML string. The XML string is stored in the *paralist[3]* parameter of the **of_execinterface** function.

The session content in the XML format will be like this.

```
<?xml version='1.0' encoding='utf-8'?>
<sessions>
  <error>
    <code>0</code>
    <text></text>
  </error>
  <sever name='192.0.3.183' port='5000'>
    <session name='2079839564'>
      <client>192.0.1.223</client>
      <application-name>sales</application-name>
      <user>userA</user>
      <state>0</state>
    </session>
    <session name='1044430361'>
      <client>192.0.3.183</client>
      <application-name>acf</application-name>
      <user>user1</user>
      <state>0</state>
    </session>
  </sever>
</sessions>
```

Method execution results are recorded in the **error** node.

Two parameters, *code* and *text*, are contained in the **error** node. The *code* parameter shows the execution result and the *text* parameter records the detail error information. If the execution result is not 0, only the **error** node is listed in the XML.

Values for the *code* parameter*:*

0*:* function succeeds in execution.

-1: function fails to execute.

-2: the specified PowerServer is not found.

-3: the specified application is not found.

Code example

```
Any la_l[]
Long lRet
...
la_l[1] = ls_servername
la_l[2] = ls_appname
la_l[3] = ls_sessioncontent

lRet = loadDBList.of_ExecInterface ("getAllSessions", la_l) //result is returned to
 la_l[3]
```

5.1.3.2 Syntax for EJB component

For Java:

io_ejb.regstring (String *serverName*)

io_ejb.regstring (String *appName*)

io_ejb.Invokeretstring (long *objid*, "**getAllSessions**", true, ref string *retval*)

Table 5.5:

Argumen	Description
io_ejb	A reference of Appeon EJBobject.
serverNam	The name of the PowerServer in which the active sessions are created. The value should be IP address or IP address with port number, for example, 192.0.0.123, or 192.0.0.123:8080, and separated with semicolons (;) if there are multiple servers (for example in a PowerServer cluster). If *serverName* is NULL or an empty string (""), **getAllSessions** returns the information of active sessions opened for the specified application in all servers in a PowerServer cluster (you will need to configure the cluster in AEM first).
appNam	The name of the application that is deployed to the specified PowerServer. If *appName* is NULL or empty string (""), **getAllSessions** returns the information of active sessions opened for all applications in the specified PowerServer.
objid	The handle to the component method.
retval	The return value of **getAllSessions** function: the session content in XML format. This parameter must be cleared, but need not to be assigned with values, as it is used to hold the return value.

Return value of Invokeretstring

Returns 1 if the **Invokeretstring** function succeeds and one of the following negative values if an error occurs.

-1: function fails to execute

-2: the specified PowerServer is not found

-3: the specified application is not found

Return value of getAllSessions

Returns an XML string. The XML string is stored in the *retval* parameter of **Invokeretstring** function.

The session content in the XML format will be like this.

```
<?xml version='1.0' encoding='utf-8'?>
<sessions>
  <error>
    <code>0</code>
    <text></text>
  </error>
  <sever name='192.0.3.183' port='5000'>
    <session name='2079839564'>
      <client>192.0.1.223</client>
      <application-name>sales</application-name>
      <user>userA</user>
      <state>0</state>
    </session>
    <session name='1044430361'>
      <client>192.0.3.183</client>
      <application-name>acf</application-name>
      <user>user1</user>
      <state>0</state>
    </session>
  </sever>
</sessions>
```

Method execution results are recorded in the **error** node.

Two parameters, *code* and *text*, are contained in the **error** node. The *code* parameter shows the execution result and the *text* parameter records the detail error information. If the execution result is not 0, only the **error** node is listed in the XML.

Values for the *code* parameter*:*

0*:* function succeeds in execution.

-1: function fails to execute.

-2: the specified PowerServer is not found.

-3: the specified application is not found.

Code example

Note that **invokeretstring** function is called because the return value is a string.

```
string ls_msg
string retval01
...
io_ejb.regstring (ls_servername)
io_ejb.regstring (ls_appname)
```

```
ls_msg = io_ejb.invokeretstring (il_bean1, "getAllSessions", true, ref retval01) //
result is returned to retval01
```

5.1.3.3 Syntax for Web Service

getAllSessions (string *serverName*, string *appName*)

Table 5.6:

Argumen	Description
serverNam	The name of the PowerServer in which the active sessions are created. The value should be IP address or IP address with port number, for example, 192.0.0.123, or 192.0.0.123:8080, and separated with semicolons (;) if there are multiple servers (for example in a PowerServer cluster). If *serverName* is NULL or an empty string (""), **getAllSessions** returns the information of active sessions opened for the specified application in all servers in a PowerServer cluster (you will need to configure the cluster in AEM first).
appName	The name of the application that is deployed to the specified PowerServer. If *appName* is NULL or empty string (""), **getAllSessions** returns the information of active sessions opened for all applications in the specified PowerServer.

Return value

XML string. For example,

```
<?xml version='1.0' encoding='utf-8'?>
<sessions>
  <error>
    <code>0</code>
    <text></text>
  </error>
  <sever name='192.0.3.183' port='5000'>
    <session name='2079839564'>
      <client>192.0.1.223</client>
      <application-name>sales</application-name>
      <user>userA</user>
      <state>0</state>
    </session>
    <session name='1044430361'>
      <client>192.0.3.183</client>
      <application-name>acf</application-name>
      <user>user1</user>
      <state>0</state>
    </session>
  </sever>
</sessions>
```

Method execution results are recorded in the **error** node.

Two parameters, *code* and *text*, are contained in the **error** node. The *code* parameter shows the execution result and the *text* parameter records the detail error information. If the execution result is not 0, only the **error** node is listed in the XML.

Values for the *code* parameter*:*

0*:* function succeeds in execution.

-1: function fails to execute.

-2: the specified PowerServer is not found.

-3: the specified application is not found.

Code example

```
//--connected to the web service
……
string ls_return, ls_server, ls_app
ls_server = sle_1.text
ls_app = sle_2.text
ls_return = ieon_ws.getAllSessions (ls_server, ls_app)
```

5.1.4 getSessionByID

Description

getSessionByID returns the detail information of the specified session with XML format.

5.1.4.1 Syntax for .NET component

proxyobject.of_execinterface ("**getSessionByID**", ref any *paralist[]*)

Table 5.7:

Argumen	Description
proxyobject	The instance of the proxy object generated for AppeonDotNetComponent.
paralist[]	Arrays of Any type. Specifies the parameter arrays for the component function. *paralist[1]* - The ID of the session. Only one ID is allowed. If this parameter is NULL or an empty string (""), in .NET, the current application session will be used as the default value. *paralist[2]* - The return value of **getSessionByID** function: the session content in XML format. This parameter must be declared, but need not to be assigned with values, as it is used to hold the return value.

Return value of of_execinterface

Returns 1 if the **of_execinterface** function succeeds and one of the following negative values if an error occurs.

-1: function fails to execute

-2: the specified session ID is not found

Return value of getSessionByID

Returns an XML string. The XML string is stored in the *paralist[2]* parameter of the **of_execinterface** function.

The session content in the XML format will be like this.

```
<?xml version='1.0' encoding='utf-8'?>
<sessions>
  <error>
    <code>0</code>
    <text></text>
  </error>
  <sever name='192.0.3.183' port='5000'>
    <session name='1044430361'>
      <client>192.0.3.183</client>
      <application-name>acf</application-name>
```

```
      <user>user1</user>
      <state>0</state>
    </session>
  </sever>
</sessions>
```

Method execution results are recorded in the **error** node.

Two parameters, *code* and *text*, are contained in the **error** node. The *code* parameter shows the execution result and the *text* parameter records the detail error information. If the execution result is not 0, only the **error** node is listed in the XML.

Values for the *code* parameter*:*

0*:* function succeeds in execution.

-1: function fails to execute.

Code example

```
Any la_l[]
Long lRet
...
la_l[1] = ls_sessionconid
la_l[2] = ls_sessioncontent

lRet = loadDBList.of_ExecInterface ("getSessionByID", la_l) //result is returned to
 la_l[2]
```

5.1.4.2 Syntax for EJB component

io_ejb.regstring (String *SessionconID*)

io_ejb.Invokeretstring (long *objid*, "**getSessionByID**", true, ref string *retval*)

Table 5.8:

Argumen	Description
io_ejb	A reference of Appeon EJBobject.
SessionconID	The ID of the session. Only one ID is allowed. If this parameter is NULL or an empty string (""), in Java, no value will be used as the default value.
objid	The handle to the component method.
retval	The return value of **getSessionByID** function: the session content in XML format. This parameter must be declared, but need not to be assigned with values, as it is used to hold the return value.

Return value of Invokeretstring

Returns 1 if the **Invokeretstring** function succeeds and one of the following negative values if an error occurs.

-1: function fails to execute

-2: the specified session ID is not found

Return value of getSessionByID

Returns an XML string. The XML string is stored in the *retval* parameter of **Invokeretstring** function.

The session content in the XML format will be like this.

```
<?xml version='1.0' encoding='utf-8'?>
<sessions>
  <error>
    <code>0</code>
    <text></text>
  </error>
  <sever name='192.0.3.183' port='5000'>
    <session name='1044430361'>
      <client>192.0.3.183</client>
      <application-name>acf</application-name>
      <user>user1</user>
      <state>0</state>
    </session>
  </sever>
</sessions>
```

Method execution results are recorded in the **error** node.

Two parameters, *code* and *text*, are contained in the **error** node. The *code* parameter shows the execution result and the *text* parameter records the detail error information. If the execution result is not 0, only the **error** node is listed in the XML.

Values for the *code* parameter*:*

0*:* function succeeds in execution.

-1: function fails to execute.

Code example

Note that **invokeretstring** function is called because the return value is a string.

```
string ls_msg
string retval01
...
io_ejb.regstring (ls_sessionconid)

ls_msg = io_ejb.invokeretstring (il_bean1, "getSessionByID", true, ref retval01) //
result is returned to retval01
```

5.1.4.3 Syntax for Web Service

getSessionByID (string *sessionid*)

Table 5.9:

Argumen	Description
sessionid	The ID of the session. Only one ID is allowed. If this parameter is NULL or an empty string (""), in .NET, the current application session will be used as the default value.

Return value

XML string. For example,

```
<?xml version='1.0' encoding='utf-8'?>
<sessions>
  <error>
    <code>0</code>
    <text></text>
  </error>
  <sever name='192.0.3.183' port='5000'>
    <session name='1044430361'>
```

```
      <client>192.0.3.183</client>
      <application-name>acf</application-name>
      <user>user1</user>
      <state>0</state>
    </session>
  </sever>
</sessions>
```

Method execution results are recorded in the **error** node.

Two parameters, *code* and *text*, are contained in the **error** node. The *code* parameter shows the execution result and the *text* parameter records the detail error information. If the execution result is not 0, only the **error** node is listed in the XML.

Values for the *code* parameter*:*

0: function succeeds in execution.

-1: function fails to execute.

Code example

```
//--connected to the web service
……
string ls_return, ls_session
ls_session= sle_3.text
ls_return = ieon_ws.getSessionByID (ls_session)
```

5.1.5 getSessionCount

Description

With getSessionCount method, you can get the following three types of information.

- The total number of active sessions opened for the specified application in the specified PowerServer.
- The total number of active sessions in a specified PowerServer.
- The total number of sessions opened for the specified application in a PowerServer cluster. To get the number of sessions in a PowerServer cluster, you need to first configure the cluster in AEM.

Usage

By using the getSessionCount method, you can easily get the total number of active sessions in a specified PowerServer using PowerBuilder code and apply the information in other open interfaces, such as KillSession, to manage the sessions. For example, you can first call getSessionCount and then call KillSession to make the deployed application kill all sessions in PowerServer when there are up to 100 active sessions in the server.

5.1.5.1 Syntax for .NET component

proxyobject.of_execinterface ("**getSessionCount**", ref any *paralist[]*)

Table 5.10:

Argumen	Description
proxyobject	The instance of the proxy object generated for AppeonDotNetComponent.

Argumen	Description
paralist[]	Arrays of Any type. Specifies the parameter arrays for the component function. *paralist[1]* - The name of the PowerServer in which the active sessions are created. The value should be IP address or IP address with port number, for example, 192.0.0.123, or 192.0.0.123:8080, and separated with semicolons (;) if there are multiple servers (for example in a PowerServer cluster). If *paralist[1]* is NULL or an empty string (""), **getSessionCount** returns the number of the active sessions in a PowerServer cluster. *paralist[2]* - The name of the application that is deployed to the PowerServer for which you want to know how many sessions are opened. If *paralist[2]* is NULL or empty string (""), **getSessionCount** returns the total number of active sessions in the specified PowerServer.

Return value of of_execinterface

Returns 0 if **of_execinterface** function succeeds and one of the following negative values if an error occurs.

-1: function fails to execute

-2: the specified PowerServer is not found

-3: the specified application is not found

Return value of getSessionCount

Returns the number of active sessions if **getSessionCount** function succeeds and returns an empty string if an error occurs.

The return value is stored in the *ReturnValue* property of the proxy object.

Code example

```
Any la_l[]
Long lRet
...
la_l[1] = ls_servername
la_l[2] = ls_appname

lRet = loadDBList.of_ExecInterface ("getSessionCount", la_l) //result is returned
 to loadDBList.ReturnValue
```

5.1.5.2 Syntax for EJB component

io_ejb.regstring (String *serverName*)

io_ejb.regstring (String *appName*)

io_ejb.Invokeretint (long *objid*, "**getSessionCount**", true, ref integer *retval*)

Table 5.11:

Argumen	Description
io_ejb	A reference of Appeon EJBobject.
serverName	The name of the PowerServer in which the active sessions are created. The value should be IP address or IP address with port number, for example, 192.0.0.123, or

Argumen	Description
	192.0.0.123:8080, and separated with semicolons (;) if there are multiple servers (for example in a PowerServer cluster). If *serverName* is NULL or an empty string (""), **getSessionCount** returns the number of the active sessions in a PowerServer cluster.
appNam	The name of the application that is deployed to the PowerServer for which you want to know how many sessions are opened. If *appName* is NULL or empty string (""), **getSessionCount** returns the total number of active sessions in the specified PowerServer.
objid	The handle to the component method.
retval	The return value of **getSessionCount** function: the number of active sessions opened for the specified application in the specified PowerServer. This parameter must be declared, but need not to be assigned with values, as it is used to hold the return value.

Return value of Invokeretint

Returns 0 if **Invokeretint** function succeeds and one of the following negative values if an error occurs.

-1: function fails to execute

-2: the specified PowerServer is not found

-3: the specified application is not found

Return value of getSessionCount

Returns the number of active sessions opened for the specified application in the specified PowerServer if **getSessionCount** function succeeds or returns an empty string if an error occurs.

The return value is stored in the *retval* parameter of **Invokeretint** function.

Code example

Note **invokeretint** function is called because the return value is an integer.

```
string ls_msg
int retval01
...
io_ejb.regstring (ls_servername)
io_ejb.regstring (ls_appname)

ls_msg = io_ejb.invokeretint (il_bean1, "getSessionCount", true, ref retval01) //
result is returned to retval01
```

5.1.5.3 Syntax for Web Service

getSessionCount (string *serverName*, string *appName*)

Table 5.12:

Argumen	Description
serverNam	The name of the PowerServer in which the active sessions are created. The value should be IP address or IP address with port number, for example, 192.0.0.123, or

Argumen	Description
	192.0.0.123:8080, and separated with semicolons (;) if there are multiple servers (for example in a PowerServer cluster). If *serverName* is NULL or an empty string (""), **getSessionCount** returns the number of the active sessions in a PowerServer cluster (you will need to configure the cluster in AEM first).
appName	The name of the application that is deployed to the PowerServer for which you want to know how many sessions are opened. If *appName* is NULL or empty string (""), **getSessionCount** returns the total number of active sessions in the specified PowerServer.

Return value

Long.

Returns the number of sessions in the specified PowerServer if the function succeeds or returns -1 if an error occurs.

Code example

```
//-- connected with the web service
……
Long ll_return
String ls_server, ls_app
ls_server = sle_1.text
ls_app = sle_2.text
ll_return = ieon_ws.getSessionCount (ls_server, ls_app)
```

5.1.6 getServerVersion

Description

getServerVersion gets the version number of PowerServer.

5.1.6.1 Syntax for .NET component

proxyobject.of_execinterface ("**getServerVersion**")

Table 5.13:

Argumen	Description
proxyobject	The instance of the proxy object generated for AppeonDotNetComponent.

Return value of of_execinterface

Returns 0 if it succeeds and -1 if it fails.

Return value of getServerVersion

String. Returns the PowerServer version number. The return value is stored in the *ReturnValue* property of the proxy object.

Code example

```
appeondotnetcomponent  ieon_dotnet
long ll_return
ieon_dotnet = create appeondotnetcomponent
ieon_dotnet.ComponentType ='4'
ieon_dotnet.TypeLib = ''
ieon_dotnet.ClassDescript = ''
```

```
ll_return = ieon_dotnet.of_execinterface ("getServerVersion") //result is returned
 to ieon_dotnet.returnvalue
if ll_return = 0 then
     mle_1.text = string (ieon_dotnet.returnvalue)
else
     messagebox ("Failed", ll_return)
end if
```

5.1.6.2 Syntax for EJB component

io_ejb.Invokeretstring (long *objid*, "**getServerVersion**", true, ref string *retval*)

Table 5.14:

Argumen	Description
io_ejb	A reference of Appeon EJBobject.
objid	The handle to the component method.
retval	The return value of **getServerVersion** function: the version number of the specified PowerServer. This parameter must be declared, but need not to be assigned with values, as it is used to hold the return value.

Return value of Invokeretstring

Returns 0 if it succeeds and -1 if it fails.

Return value of getServerVersion

String. Returns the version number of PowerServer. The return value is stored in the *retval* property of the **Invokeretstring** function.

Code example

Note **invokeretstring** function is called because the return value is a string.

```
//Connects with EJB and returns the instance handle il_bean
string ls_return, ls_msg
ls_return = ieon_ejb.invokeretstring (il_bean, "getServerVersion", true, ref
 ls_msg)
//returns an empty string to ls_return if it succeeds, and returns the PowerServer
 version number to ls_msg
```

5.1.6.3 Syntax for Web Service

getServerVersion ( )

Return value

String. Returns the version number of PowerServer if it succeeds and returns an empty string if an error occurs.

Code example

```
//-- connected with the web service
……
String ls_return
ls_return = ieon_ws.getServerVersion()
```

5.1.7 killAllSessions

Description

killAllSessions kills all active sessions in a PowerServer or a PowerServer cluster and rolls back all associated transactions. To kill all sessions in a PowerServer cluster, you need to first configure the cluster in AEM.

5.1.7.1 Syntax for .NET component

proxyobject.of_execinterface ("**killAllSessions**", ref any *paralist[]*)

Table 5.15:

Argumen	Description
proxyobject	The instance of the proxy object generated for AppeonDotNetComponent.
paralist[]	Arrays of Any type. Specifies the parameter arrays for the component function. *paralist[1]* - The name of PowerServer for which you want to kill all sessions. The value should be IP address or IP address with port number, for example, 192.0.0.123, or 192.0.0.123:8080, and separated with semicolons (;) if there are multiple servers (for example in a PowerServer cluster). If *paralist[1]* is NULL or empty string (""), it kills all sessions and rolls back all associated transactions in a PowerServer cluster.

Return value of of_execinterface

Returns 0 if it succeeds and -1 if it fails.

Return value of killAllSessions

Boolean.

It returns true if it succeeds and false if it fails. The return value is stored in the *ReturnValue* property of the proxy object.

Code example

```
Any la_l[]
Long lRet
...
la_l[1] = ls_servername

lRet = loadDBList.of_ExecInterface ("killAllSessions", la_l) //result is returned
 to loadDBList.ReturnValue
```

5.1.7.2 Syntax for EJB component

io_ejb.regstring (String *serverName*)

io_ejb.Invokeretbool (long *objid*, "**killAllSessions**", true, ref boolean *retval*)

Table 5.16:

Argumen	Description
io_ejb	A reference of Appeon EJBobject.
serverName	The name of PowerServer for which you want to kill all sessions. The value should be IP address or IP address with port number, for example, 192.0.0.123, or 192.0.0.123:8080, and separated with semicolons (;) if there are multiple servers (for example in a PowerServer cluster).

Argumen	Description
	If *serverName* is NULL or empty string (""), it kills all sessions and rolls back all associated transactions in a PowerServer cluster.
objid	The handle to the component method.
retval	The return value of **killAllSessions** function: true if it succeeds and false if it fails. This parameter must be declared, but need not to be assigned with values, as it is used to hold the return value.

Return value of Invokeretbool

Returns 0 if it succeeds and -1 if it fails.

Return value of killAllSessions

Boolean.

It returns true if it succeeds and false if it fails. The return value is stored in the *retval* property of the **Invokeretbool** function.

Code example

Note that **invokeretbool** function is called because the return value is a boolean.

```
string ls_msg
boolean retval01
...
io_ejb.regstring (ls_servername)

ls_msg = io_ejb.invokeretbool (il_bean1, "killAllSessions", true, ref retval01) //
result is returned to retval01
```

5.1.7.3 Syntax for Web Service

killAllSessions (String *as_server*)

Table 5.17:

Argumen	Description
as_server	The name of PowerServer for which you want to kill all sessions. The value should be IP address or IP address with port number, for example, 192.0.0.123, or 192.0.0.123:8080, and separated with semicolons (;) if there are multiple servers (for example in a PowerServer cluster). If *as_server* is NULL or empty string (""), it kills all sessions and rolls back all associated transactions in a PowerServer cluster.

Return value

Returns 1 if it succeeds and -1 if it fails.

Code example

```
//-- connected with the web service
……
string ls_return,ls_server,ls_app
ls_server = sle_1.text
ls_app = sle_2.text
ls_return = ieon_ws.killallsessions(ls_server)
```

```
mle_1.text = ls_return
```

5.1.8 killSessions

Description

killSessions kills the specified session(s) in a PowerServer or a PowerServer cluster and rolls back the associated transactions. To kill sessions in a PowerServer cluster, you need to first configure the cluster in AEM. The mobile session IDs listed in AEM are not the real session IDs. You can view source to get the real mobile session IDs.

5.1.8.1 Syntax for .NET component

proxyobject.of_execinterface ("**killSessions**", ref any *paralist[]*)

Table 5.18:

Argumen	Description
proxyobject	The instance of the proxy object generated for AppeonDotNetComponent.
paralist[]	Arrays of Any type. Specifies the parameter arrays for the component function. *paralist[1]* - The ID of sessions which you want to kill. If there are multiple sessions, separate them with semicolons (;).

Return value of of_execinterface

Returns 0 if it succeeds and -1 if it fails.

Return value of killSessions

Boolean.

It returns true if it succeeds and false if it fails. The return value is stored in the *ReturnValue* property of the proxy object.

Code example

```
Any la_l[]
Long lRet
...
la_l[1] = ls_sessionids

lRet = loadDBList.of_ExecInterface ("killSessions", la_l) //result is returned to
 loadDBList.ReturnValue
```

5.1.8.2 Syntax for EJB component

io_ejb.regstring (String *sessionIDs*)

io_ejb.Invokeretbool (long *objid*, "**killSessions**", true, ref boolean *retval*)

Table 5.19:

Argumen	Description
io_ejb	A reference of Appeon EJBobject.
sessionIDs	The ID of sessions which you want to kill. If there are multiple sessions, separate them with semicolons (;).

Argumen	Description
objid	The handle to the component method.
retval	The return value of **killSessions** function: true if it succeeds and false if it fails. This parameter must be declared, but need not to be assigned with values, as it is used to hold the return value.

Return value of Invokeretbool

Returns 0 if it succeeds and -1 if it fails.

Return value of killSessions

Boolean.

It returns true if it succeeds and false if it fails. The return value is stored in the *retval* property of the **Invokeretbool** function.

Code example

Note that **invokeretbool** function is called because the return value is a boolean.

```
string ls_msg
boolean retval01
...
io_ejb.regstring (ls_sessionids)

ls_msg = io_ejb.invokeretbool (il_bean1, "killSessions", true, ref retval01) //
result is returned to retval01
```

5.1.8.3 Syntax for Web Service

killSessions (string *sessionstring*)

Table 5.20:

Argumen	Description
sessionstring	The ID of the session(s) which you want to kill. Multiple session IDs are separated with semicolons (;).

Return value

Returns 1 if it succeeds and -1 if it fails.

Code example

```
//-- connected with the web service
……
string ls_return,ls_server,ls_app,ls_session
ls_server = sle_1.text
ls_app = sle_2.text
ls_session = sle_3.text
ls_return = ieon_ws.killsessions(ls_session)
mle_1.text = ls_return
```

5.1.9 rollbackAllTransactions

Description

rollbackAllTransactions rolls back all transactions in a PowerServer or a PowerServer cluster. To roll back all transactions in an PowerServer cluster, you need to first configure the cluster in AEM.

5.1.9.1 Syntax for .NET component

proxyobject.of_execinterface ("**rollbackAllTransactions**", ref any *paralist[]*)

Table 5.21:

Argumen	Description
proxyobject	The instance of the proxy object generated for AppeonDotNetComponent.
paralist[]	Arrays of Any type. Specifies the parameter arrays for the component function. *paralist[1]* - The name of the PowerServer that you want to kill all sessions on. The value should be IP address or IP address with port number, for example, 192.0.0.123, or 192.0.0.123:8080, and separated with semicolons (;) if there are multiple servers (for example in a PowerServer cluster). If *paralist[1]* is NULL or empty string (""), it rolls back all transactions in a PowerServer cluster.

Return value of of_execinterface

Returns 0 if it succeeds and -1 if it fails.

Return value of rollbackAllTransactions

Boolean.

It returns true if it succeeds and false if it fails. The return value is stored in the *ReturnValue* property of the proxy object.

Code example

```
Any la_l[]
Long lRet
...
la_l[1] = ls_servername

lRet = loadDBList.of_ExecInterface ("rollbackAllTransaction", la_l) //result is
 returned to loadDBList.ReturnValue
```

5.1.9.2 Syntax for EJB component

io_ejb.regstring (String *serverName*)

io_ejb.Invokeretbool (long *objid*, "**rollbackAllTransactions**", true, ref boolean *retval*)

Table 5.22:

Argumen	Description
io_ejb	A reference of Appeon EJBobject.
serverName	The name of the PowerServer that you want to kill all sessions on. The value should be IP address or IP address with port number, for example, 192.0.0.123, or 192.0.0.123:8080, and separated with semicolons (;) if there are multiple servers (for example in a PowerServer cluster).

Argumen	Description
	If *serverName* is NULL or empty string (""), it rolls back all transactions in a PowerServer cluster.
objid	The handle to the component method.
retval	The return value of **rollbackAllTransactions** function: true if it succeeds and false if it fails. This parameter must be declared, but need not to be assigned with values, as it is used to hold the return value.

Return value of Invokeretbool

Returns 0 if it succeeds and -1 if it fails.

Return value of rollbackAllTransactions

Boolean.

It returns true if it succeeds and false if it fails. The return value is stored in the *retval* property of the **Invokeretbool** function.

Code example

Note that **invokeretbool** function is called because the return value is a boolean.

```
string ls_msg
boolean retval01
...
io_ejb.regstring (ls_servername)

ls_msg = io_ejb.invokeretbool (il_bean1, "rollbackAllTransaction", true, ref
 retval01) //result is returned to retval01
```

5.2 Calling PowerServer open interfaces via EJB component

PowerServer open interfaces (methods) in JBoss, JEUS, WebLogic and WebSphere are encapsulated in a standard EJB component named **OpenInterface** that is included in the **ASInterface** package. To invoke the open interfaces in PowerBuilder, you need to follow the instructions of how to call Appeon EJBObject. Details refer to Section 2.3.6, “Calling EJB Component”.

Following is general step guide to write code in PowerBuilder to invoke the open interface.

Step 1: Load Appeon Workarounds PBL to the application.

Step 2: Deploy Appeon Bridge.

Step 3: Connect to the J2EE application server where PowerServer and Appeon Bridge are installed through Appeon EJBobject.

For example:

```
string ls_prop[5]
string ls_serurl, ls_msg
ls_serurl = "http://192.0.3.35:8000/appeonbridge/Dispatch"
ls_prop[1] = "applicationA"
ls_prop[2] =
 "javax.naming.Context.INITIAL_CONTEXT_FACTORY='com.sybase.ejb.InitialContextFactory'"
ls_prop[3] = "javax.naming.Context.PROVIDER_URL='iiop://192.0.1.249:2000'"
ls_prop[4] = "username=admin@system"
ls_prop[5] = "password=appeon249"
```

```
ls_msg = io_ejb.connectserver (ls_serurl, ls_prop)
if ls_msg <> "" then
MessageBox ("Error when connecting with the application server!", ls_msg)
return
end if
```

Step 4: Create an instance of the EJB component OpenInterface.

The JNDI name for WebSphere, WebLogic, JEUS, & JBoss is **OpenInterfaceBean**.

For example:

```
String ls_jndi, ls_home, ls_method, ls_msg
ls_jndi = "OpenInterfaceBean"
ls_msg = io_ejb.createremoteinstance (ls_jndi, ls_home, "create", ref il_bean1)
if ls_msg <> "" then
MessageBox ("Failed to create the remote instance!", ls_msg)
return
end if
```

Step 5: Invoke one or more open interfaces in the component. For example:

```
string ls_msg
int c_val
string retval01
c_val = -1
io_ejb.regstring ("")
io_ejb.regstring ("")
ls_msg = io_ejb.invokeretstring (il_bean1, "getAllSessions", true, ref retval01)
if ls_msg = "" then
messagebox ('', "retval=" + string(retval01))
else
messagebox ("", "retval=" + ls_msg)
end if
```

Step 6: Deploy the PowerBuilder application to PowerServer the same way you would deploy a normal PowerBuilder application.

The above code will take effect in the Appeon-deployed application.

5.3 Calling PowerServer open interfaces via .NET component

PowerServer open interfaces (methods) in .NET are encapsulated in a standard .NET component. To call the .NET component, Appeon provides a non-autoinstantiated NVO - AppeonDotNetComponent - as the proxy object to call the server-side component. Details refer to Section 2.3.7, “Calling .NET/COM server components (.NET only)”.

Following is general step guide to write code in PowerBuilder to invoke the open interface.

Step 1: Load Appeon Workarounds PBL to the application.

Step 2: Make sure the .NET/COM components have been registered using the **regsvr32** tool and the components have been copied to the **%appeon%/AEM/components** folder on the PowerServer machine.

Step 3: Create an instance of **AppeonDotNetComponent**. For example:

```
String servername, appname
AppeonDotNetComponent loadDBList
Any la_l[]
Long lRet
serverName = ""
```

```
appname = ""
la_l[1] = servername
la_l[2] = appname
loadDBList = Create AppeonDotNetComponent
```

Step 4: Specify the **ComponentType** and set the **TypeLib** property and the **ClassDescript** property to empty. For example:

```
loadDBList.ComponentType = "4"
loadDBList.TypeLib = ""
loadDBList.ClassDescript = ""
```

Step 5: Invoke one or more open interfaces in the component. For example:

```
lRet = loadDBList.of_ExecInterface ("getSessionCount", la_l)
If lRet = 0 Then
MessageBox ("Get session count successfully.", "Session Count:" +
 string(loadDBList.ReturnValue))
End If
```

Step 6: Deploy the PowerBuilder application to PowerServer the same way you would deploy a normal PowerBuilder application.

The above code will take effect in the Appeon-deployed application.

5.4 Calling PowerServer open interfaces via Web service

Here is the code example that shows you how to call the PowerServer open interfaces via Web service in the PowerBuilder application.

Step 1: Generate the proxy object for the Web service using the .NET Web service engine.

- URL for the open interface Web service in PowerServer installed to the Java application server (such as JBoss, WebLogic, WebSphere, or JEUS): http://*AppeonServerName*:*Port*/servlet/AppeonService?wsdl
- URL for the open interface Web service in PowerServer installed to the .NET IIS server: http://*AppeonServerName*/servlet/AppeonService.asmx

Step 2: Create the instance for the SOAP connection object and the Web service proxy object.

```
soapconnection  ieon_soap
wsappeonservice ieon_ws    //the Web service proxy object generated by the .NET
 engine
wsauthsoapheader ieon_head
long ll_return
string ls_return, ls_server, ls_app

ieon_soap = create soapconnection
ieon_head = create wsauthsoapheader
```

Step 3: Connect with the Web service, and pass in the user name and password for security authentication.

```
ll_return = ieon_soap.createinstance (ieon_ws, 'wsappeonservice')
ieon_head.username = "admin"
ieon_head.password = "admin"
ieon_ws.setauthsoapheadervalue (ieon_head)
```

Step 4: Call the PowerServer open interface.

```
ls_server = sle_1.text
ls_app = sle_2.text
ls_return = ieon_ws.getAllSessions (ls_server, ls_app)
```

6 Workarounds for Unsupported Features

Not all of the PowerBuilder features can be supported by Appeon. The unsupported features, if not modified, will be commented out in the generated application files, as a result, the code that contains the unsupported features and other code that is dependent on those unsupported features will stop working.

This chapter provides suggestions to work around the unsupported features that have functional impact on the running of the application. Some cosmetic features, such as **BorderStyle=StyleShadowBox!** property, can be simply ignored if they will not affect the application.

6.1 Objects & Controls

6.1.1 External user object

Description

The external user object is unsupported.

Workaround

Choose one of the following methods:

Method #1: Encapsulate the functions using DLLs.

Method #2: Encapsulate the functions using a user-defined OCX.

6.1.2 UserObject object

6.1.2.1 DeleteItem function

Description

DeleteItem function is unsupported in a user defined object.

Workaround

Call the DeleteItem function of controls such as ListBox, DropDownListBox, PictureListBox, DropDownPictureListBox, or ListView instead of calling the DeleteItem function of a user-defined object.

6.1.3 System Objects

6.1.3.1 Application object

6.1.3.1.1 DWMessageTitle property

Description

The DWMessageTitle is unsupported.

Workaround

Use a global variable to record the value of the DWMessageTitle property. Read and write the global variable instead of using the DWMessageTitle property in the Script.

6.1.3.1.2 FreeDBLibraries property

Description

The FreeDBLibraries property is unsupported.

Workaround

It is a useless property for a Web based application. The property can be simply commented out causing no functionality difference for the deployed application.

6.1.4 System Controls

6.1.4.1 ListView control

6.1.4.1.1 GetItemAtPointer function

Description

GetItemAtPoint is unsupported for ListView control.

Workaround

Replace the use of GetItemAtPointer by using Index argument of RightClicked since they return the same value.

6.1.4.1.2 ItemActivate event

Description

The ItemActivate event for ListView control is unsupported.

Workaround

Copy the script in the ItemActivate event to the Clicked or DoubleClicked event.

6.1.4.2 TreeView control

6.1.4.2.1 SetDropHighlight function

Description

The SetDropHightlight function for TreeView control is unsupported.

Workaround

Use other functions to highlight the item specified in SetDropHighlight as the drop target. For example, change the font of the item or modify the item label.

6.1.4.3 Tab control

6.1.4.3.1 TabPostEvent function

Description

TabPostEvent function of tab control is unsupported.

Workaround

Use TabTriggerEvent instead.

6.2 System Functions

6.2.1 DDE Server functions

Description

The DDE Server functions are unsupported.

Workaround

Use a DLL to work around this issue.

1. Encapsulate the DDE in a DLL.
2. Create a DLL proxy to call the DDE in the DLL.

6.2.2 FileLength64 function

Description

The FileLength64 function is unsupported.

Workaround

Replace the FileLength64 function with FileLength function.

6.2.3 FileReadEx function

The FileReadEx function is supported since Appeon 6.5.

Description

The FileReadEx function is unsupported in Appeon 5.x, 6.0, 6.1 and 6.2.

Workaround

Replace the FileReadEx function with FileRead function.

6.2.4 FileSeek64 function

Description

The FileSeek64 function is unsupported.

Workaround

Replace the FileSeek64 function with FileSeek function.

6.2.5 FileWriteEx function

The FileWriteEx function is supported since Appeon 6.5.

Description

The FileWriteEx function is unsupported in Appeon 5.x, 6.0, 6.1 and 6.2.

Workaround

Replace the FileWriteEx function with FileWrite function.

6.2.6 FromAnsi function

Description

The FromAnsi function is unsupported.

Workaround

Replace the FromAnsi function with String function.

Example

The original code:

```
FromAnsi ( blob )
```

The modified code in PowerBuilder 9:

```
String(blob)
```

The modified code in PowerBuilder 10:

```
String(blob, EncodingANSI!)
```

6.2.7 FromUnicode function

Description

The FromUnicode function is unsupported.

Workaround

Replace the FromUnicode function with String function.

Example

The original code:

```
FromUnicode(blob)
```

The modified code:

```
String(blob)
```

6.2.8 Garbage Collection functions

Description

The Garbage Collection functions (GarbageCollect, GarbageCollectGetTimeLimit & GarbageCollectSetTimeLimit) are unsupported.

Workaround

Appeon has its own mechanism to delete unused objects and classes. This function can be simply commented out, causing no loss to the deployed application.

6.2.9 GetLibraryList function

Description

The GetLibraryList function is unsupported.

Workaround

Create a new table holding all the library information in the database. Use this table to check the information in the library instead of using the GetLibraryList function.

6.2.10 Help functions

Description

Using the Help functions such as ShowHelp or ShowPopupHelp to display a HLP file or a CHM (HTML Help) file is unsupported.

Workaround

To enable you to display a HLP file or a CHM file, replace the Help functions by applying a StaticHyperLink, PictureHyperLink or HyperLinkToURL:

Step 1: Upload the Help file to your Web Server.

Step 2: Create a StaticHyperLink or PictureHyperLink control in the application and set the link of the control to the URL of the Help file. You can also apply the HyperLinkToURL function in the application for accessing the URL of the Help file.

Example

This example uses PowerServer Help as the HLP file. First upload the HLP file to the Web Server (in this example PowerServer Help is located in http://www.appeon.com/support).

Workaround #1: Open the HLP or CHM file with a StaticHyperLink control.

Create a StaticHyperLink control in the application. Add the PowerServer Help URL (http://www.appeon.com/support) to the URL property of StaticHyperLink control.

Workaround #2: Apply the HyperLinkToURL function to access the URL of the Help file.

Add the following code to the relevant location in the application:

```
Inet internet
Internet = create Inet
internet.HyperLinktoURL("http://www.appeon.com/support/manuals")
```

6.2.11 HyperLinkToURL function

Description

While the HyperLinkToURL function is supported, it cannot link to the URL of an Appeon application. This is because an Internet Explorer process only supports one Appeon application at a time. Executing the HyperLinkToURL function from one Appeon application to open another will cause Internet Explorer to shut down.

Workaround

Method 1: Using the Run function to open the second application

Step 1: Replace the call to the HyperLinkToURL function with a call to the Run system function. Comment out the script in the HyperLinkToURL button and add the following script:

```
run("IEXPLORE.EXE http://192.0.2.189:8080/b237293/",Maximized!);
```

In the code example, "http://192.0.2.189:8080/b237293" can be replaced with the URL of an Appeon application.

Step 2: At the Web server, copy the IEXPRLORE.EXE to *C:\WINNT\system* or *C:\WINNT \system32*. Alternatively, add the directory *C:\Program Files\Internet Explorer* in the PATH variable.

Method 2: Using the PictureHyperLink URL property to open the second application

Step 1: Add a PictureHyperLink control to the first application, from which the second application will be opened.

Step 2: Specify the URL of the second application as the URL of the PictureHyperLink control.

6.2.12 LibraryDirectory function

Description

The LibraryDirectory PowerScript function is unsupported.

Workaround

The LibraryDirectory function is used to get a list of all objects or just objects of a specified type in a PowerBuilder library.

Case #1

If the LibraryDirectory function is used in other unsupported Library functions, there is no workaround available. Comment out the script related to the Library functions and the relevant functionality will be lost.

Case #2

If the LibraryDirectory function is used to get the list of all objects in the PowerBuilder Library and check whether a particular object exists, the workaround steps are:

Step 1: Create a table in the database to store the object names of the PowerBuilder library that is passed to the LibraryDirectory function.

Step 2: Use a SQL statement to check whether a particular object exists in the table.

6.2.13 PopulateError function

Description

The PopulateError function is unsupported.

Workaround

Appeon has its own error-handling mechanism. This function can be directly commented out, causing no loss to the deployed application.

6.2.14 Registry function

Description

System registry functions can read and write registry entries, keys, and values on a Windows PC. But unlike Windows system, iOS system has no such registry, so these functions cannot be directly executed in iOS system.

Workaround

Appeon offers an option of "Appeon emulation registry" in AEM to allow the mobile application to read and write the mock registry file stored in the PowerServer database. For detailed information, please refer to **PowerServer Configuration Guide for .NET** | **AEM User Guide** | **Application** | **PB Features** | **Registry Mode**.

Note: Mobile applications can only read Appeon emulation registry.

6.2.15 Shared Object functions

Description

Shared Object functions like SharedObjectGet, SharedObjectRegister are unsupported by Appeon, so they cannot be applied to show the process of an event with a progress bar, or control the progress of two different processes.

Workaround

The following is the workaround for showing the process of an event to the users:

Before the execution of the event, prompt a message box at the client to inform the user that the event (such as the retrieve of a large amount of data) is taking place and the event may take relatively long.

6.2.16 SignalError function

Description

The SignalError function is unsupported.

Workaround

Appeon has its own error-handling mechanism. This function can be directly commented out, causing no functionality loss to the deployed application.

6.3 PowerScript Reference

6.3.1 PowerScript Topics

6.3.1.1 Calling functions and events

6.3.1.1.1 Overriding system function

Description

Overriding system functions is unsupported.

Workaround

Create a user-defined function and rewrite the user code in the system function into the user-defined function.

Example

In the original application, the SetTransObject function of the *u_dw* object contains the following user code:

```
trans_current = atrans_current return super::settransobject(itrans_current)
```

In the converted Web application, remove the preceding user code from the SetTransObject function, then create a user-defined function *uf_settransobject* and rewrite the following code:

```
itrans_current = atrans_current settransobject(itrans_current)
```

When the SetTransObject function of the *u_dw* object is called in the Web application, the user code is executed in the newly created *uf_settransobject* function.

6.3.1.1.2 Passing arguments to functions and events

Duplicate arguments for a function

Description

Repetitively referring objects as arguments for a function is unsupported. Use the following workaround and example to change it into a supported format.

Workaround

Assign the repetitively referenced object to multiple different variables and pass these variables to a function.

Example

The original script:

```
w_1.wf_1(dw_1,dw_1)
```

Re-write it using the following format:

```
u_dw ldw_1, ldw_2
ldw_1 = dw_1
ldw_2 = dw_1
w_1.wf_1(ldw_1,ldw_2)
```

Passing Menu object as a reference parameter

Description

Passing Menu object as a reference parameter is unsupported.

Workaround

Pass the Menu object by value or as read-only instead of by reference. The reason is, when passing by value or as read-only, if you change the properties of the object by value or as read-only, you are changing the original object, which is the same as passing by reference.

Reference parameter

Description

The return values of functions and properties of objects cannot be directly used as reference parameters for functions.

Workaround

Follow the steps below to work around this issue:

Step 1: Create a variable and assign the return value of a function or the property of an object to the variable.

Step 2: Call the variable where the return value or the property is called in the original code.

Step 3: Assign the return value of calling the variable to the original property or object.

Code example

The original code

```
poptags(theobject.item[theitem])
```

Modified to

```
menu lm_menu
lm_menu = theobject.item[theitem]
poptags(lm_menu)
```

6.3.1.2 Declarations

6.3.1.2.1 Shared variables

Description

Shared variables are unsupported. For example, In the *pfcmain.pbl* of a PFC application, the *pfc_u_dw* uses a shared variable *snv_property* with its type being *n_cst_dwsrv_property*. The *snv_property* is unsupported.

Workaround

Follow the steps below to work around this issue.

Step 1: Change the variable *snv_property* as an instance variable of *n_cst_appmanager* (usually declared as gnv).

Step 2: Open the *pfc_u_dw*, and replace all the "snv_property" with the "gnv_app.snv_property".

Note

Since the parent code is changed, when deploying a new PFC, the same modification needs to be done again.

Other shared variables can also be worked around in this way.

6.3.1.3 Language basics

6.3.1.3.1 Null values

Description

The Null value calculation in Appeon is quite different from that in PowerBuilder. This is caused by the different calculation methods used in PowerScript and JavaScript.

Workaround

When using expressions with Null value calculations, you should add conditional statements to ensure that you get the correct result.

Example #1

A and/or B is/are likely to carry a Null value in the following assignment:

```
C = A + B
```

Re-write it using the following format:

```
IF IsNull(A) or IsNull(B) THEN
 SetNull(C)
ELSE
 C = A + B
END IF
```

Example #2

Another example for relational operations:

```
IF A = B then
 MessageBox ("Return Value","True")
ELSE
 MessageBox ("Return Value","False")
END IF
```

Re-write it using the following format:

```
IF IsNull(A) or IsNull(B) then
 MessageBox ("Return Value", "False")
ELSEIF A = B then
 MessageBox ("Return Value", "True")
ELSE
 MessageBox ("Return Value", "False")
END IF
```

Note

The following table shows the different return values that PowerBuilder and Appeon will produce when an expression contains at least one null value. In these examples, the values of variables A and B are both null:

Expressions	Return Value in PowerBuilder	Return Value in Appeon
A+1	Null	1
A+B	Null	Null
A*B	Null	0
A=1 (relational)	Null	False
A<>1	Null	True
NOT (A=1)	Null	True
A=A (relational)	Null	True
A=B	Null	True
IsNull(A=1)	True	False

6.3.2 PowerScript Statements

6.3.2.1 GOTO statement

Description

PowerScript GOTO statements and Label are unsupported. Using GOTO is not recommended in structured programming.

Workaround

Analyze the code that uses GOTO and re-write the code in a structured way by applying IF ... THEN ... statements.

If the statement that the GOTO label is associated with has a return value, place the statement in a user function, and in place of the GOTO statement, call the user function.

If the statement that the GOTO label is associated with has no return value, place the statement directly in place of the GOTO statement.

Example

The original script:

```
IF sle_1.text = "" THEN GOTO hide_sle_1
sle_1.text = ""
hide_sle_1:
sle_1.visible = false
MessageBox ("","SingleLineEdit sle_1 is cleared and hidden.")
```

Re-write it using IF ... THEN... statement:

```
IF sle_1.text = "" THEN
sle_1.visible = false
MessageBox ("","SingleLineEdit sle_1 is cleared and hidden.")
ELSE
sle_1.text = ""
sle_1.visible = false
MessageBox ("","SingleLineEdit sle_1 is cleared and hidden.")
END IF
```

Re-write it using CHOOSE CASE statement:

```
CHOOSE CASE sle_1.text
CASE ""
sle_1.visible = false
MessageBox ("","SingleLineEdit sle_1 is cleared and hidden.")
CASE ELSE
sle_1.text = ""
sle_1.visible = false
MessageBox ("","SingleLineEdit sle_1 is cleared and hidden.")
END CHOOSE
```

6.3.3 SQL Statements

6.3.3.1 Stored procedure with Null output

Description

It is currently unsupported to use NULL value in an OUTPUT parameter when declaring a stored procedure.

Workaround

Set a variable and set it to NULL. Then use the variable in the OUTPUT parameter to provide the same functionality.

Example

Original code:

```
// Declare the procedures
Declare SPgetseniva procedure for SPgetseniva &
@codpes = :l_usp_codpes, &
@nomsis = :s_usp_systemname, &
@staobt = :s_usp_systemstatus,&
@numseniva = null output ;
```

Modified code:

```
integer li_return
SetNull(li_return)
// Declare the procedures
Declare SPgetseniva procedure for SPgetseniva &
@codpes = :l_usp_codpes, &
@nomsis = :s_usp_systemname, &
@staobt = :s_usp_systemstatus,&
@numseniva = :li_return output ;
```

6.3.4 PowerScript Events

6.3.4.1 Help event

Description

The Help event is unsupported.

Workaround

Copy the script of the Help event to the Key event. In the Key event, use the KeyDown function to check whether the user has pressed the F1 key and if the KeyDown function returns true, execute the script that is originally in the Help event.

6.3.4.2 Other event

Description

The Other event is unsupported.

Workaround

Move the script in the Other event to the supported events.

6.4 DataWindow Reference

6.4.1 DataWindow operators

6.4.1.1 DataWindow operator precedence

Description

The AND and OR operators in a DataWindow expression have the same precedence in PowerBuilder, but in Appeon, the AND operator has higher precedence.

Workaround

When there are both AND and OR operators in a DataWindow expression, you should use parentheses to get the correct precedence effect.

Example

The following code examples are for a DataWindow expression that sets the column text color. The OR operator will be evaluated first in PowerBuilder, but in Appeon, the following script will evaluate the AND operator first.

Original code:

```
If(Left(GetText(), 1) = 'V' OR Left(GetText(), 1) = 'A' AND Mod(GetRow(), 2) = 1,
 236, 243433) // incorrect
```

To have the OR operator evaluated first, add a pair of parentheses to the OR expression:

```
If((Left(GetText(), 1) = 'V' OR Left(GetText(), 1) = 'A') AND Mod(GetRow(), 2) = 1,
 236, 243433) // correct
```

6.4.2 DataWindow Object Properties

6.4.2.1 Retrieve.AsNeeded

Description

The Retrieve.AsNeeded property of DataWindow object is unsupported.

Workaround

Write script to have the data displayed in batches instead of at one time. The detailed steps are:

Step 1: Create a user object. The user object contains four Picture buttons. The Picture buttons represents First page, Previous page, Next page and Last page respectively.

Step 2: Place the user object in the window that contains the DataWindow in which the large quantity of data will be displayed.

Step 3: Define window functions and modify the original scripts to have them work along with the user object to provide the functionality of displaying data in batches.

6.4.3 DataWindow control

6.4.3.1 Functions

6.4.3.1.1 CanUndo function

Description

CanUndo function is unsupported.

Workaround

Replace the CanUndo function with ModifiedCount function.

Example

The original code:

```
if dw_1.CanUndo() Then
dw_1.Undo()
end if
```

The modified code:

```
if dw_1.ModifiedCount() > 0 Then
dw_1.Undo()
End if
```

6.4.3.1.2 DBCancel function

Description

DBCancel function is unsupported.

Workaround

Appeon has its own mechanism to retrieval data. This function can be simply commented out.

Functionality difference

All the retrieved data will be returned at the same time.

6.4.3.1.3 GenerateResultSet function

Description

The ResultSet system object and GenerateResultSet method are unsupported.

Workaround

To work around the GenerateResultSet method, we have the following two methods:

Method #1: Use GetFullState to retrieve data from a DataWindow and then use SetFullState to apply the blob returned from GetFullState to another DataWindow.

Method #2: Use the datawindow.data property to retrieve data from a DataWindow into a string and insert data into the DataWindow from the string by ImportString.

6.4.3.1.4 GetTrans function

Description

The DataWindow GetTrans function is unsupported.

Workaround

Comment out the unsupported script. Instead use the SetTransObject connection method, to assign a programmer-specified transaction object or a global transaction object called SQLCA to a DataWindow control or DataStore.

To use SetTransObject, write code that performs the following tasks:

1. Set up the transaction object by assigning values to its fields (usually in the application's Open event).
2. Connect to the database using the SQL CONNECT statement and the transaction object (in the Open event for the application or window).
3. Call SetTransObject to associate the transaction object with the DataWindow control or DataStore (usually in the window's Open event).
4. Check the return value from the Update method and follow it with an SQL COMMIT or ROLLBACK statement, as appropriate.

6.4.3.1.5 ReselectRow function

Description

The DataWindow ReselectRow method is unsupported.

Workaround #1

If the DataWindow's source table has a primary key, to work around this issue, follow the steps below:

Step 1: Use GetItem() to get the value of the primary column in the current row.

Step 2: Use SQL statement to retrieve data for the current row according to the value of the primary key.

Step 3: Use SetItem() to assign values to each column in the current row.

Step 4: Change the status of the current row to "NotModified!"

Workaround #2

Replace the ReselectRow function with the Retrieve function.

6.4.3.1.6 ResetTransObject function

Description

The ResetTransObject function of a DataWindow control or a DataStore is unsupported.

Workaround

Replace the ResetTransObject function with SetTransObject function.

Example

The original script:

```
dw_1.ResetTransObject()
```

The modified script:

```
dw_1.SetTransObject(transaction)
```

6.4.3.1.7 Scroll function

Description

The Scroll function of DataWindow control is unsupported.

Workaround

Replace the Scroll function with ScrollToRow, ScrollPriorPage or ScrollNextRow function.

Functionality difference

The event sequence of the corresponding function will be triggered.

Example

The original script:

```
dw_1.Scroll(3)
```

The modified script. Please note that the event sequence of the ScrollToRow function will be triggered.

```
dw_1.ScrollToRow(dw_1.GetRow() + 3)
```

6.4.3.1.8 SetTrans function

Description

Using SetTrans to establish a connection to the transaction object is unsupported.

Workaround

Use the SetTransObject method.

Example

The original script:

```
i = ids_main.SetTrans(itr_sql)
```

Re-write it using the following format:

```
ids_main.setTransObject(itr_sql)
```

6.4.3.2 Events

6.4.3.2.1 ScrollVertical event

Description

ScrollVertical event is unsupported.

Workaround

You can choose either of the following two ways to work around the ScrollVertical event.

To work around the ScrollVertical event for a single DataWindow

Step 1: Define a user-defined event such as ue_scrollvertical to replace ScrollVertical.

Step 2: Place the same code that you plan to put in ScrollVertical in the user-defined event.

Step 3: In the Timer event of the window that hosts the DataWindow, trigger the user-defined event periodically with the following code:

```
long ll_new_firstrow
ll_new_firstrow = long(dw_2.Object.DataWindow.firstRowOnPage) // Assuming the
 DataWindow is dw_2
if ll_new_firstrow <> il_old_first_row then
 dw_2.trigger event ue_scrollvertical()
 il_old_first_row = ll_new_firstrow
end if
```

Step 4: Call the Timer event in the Open event of the window with the following code:

```
timer(0.005)
```

To work around the ScrollVertical event for ancestor DataWindow

You can code the workaround once in the ancestor DataWindow and then apply it for as many DataWindow as you want.

Supposing the ancestor window is w_sheet and the ancestor DataWindow is u_dw:

Step 1: Start the Timer event in the Open event of w_sheet.

```
string ls_timer_interval
```

```
ls_timer_interval = &
ProfileString( gnv_app.of_getappinifile( ), "timer", "interval", "0.005" );
timer(Dec(ls_timer_interval));
```

Step 2: In the pfc_postopen event of w_sheet, add the following code for getting all the DataWindows.

```
of_sb_get_dwobjects( this.control );
```

Step 3 - Add a new function called of_sb_get_dwobjects (windowobject awo_control[]) in w_sheet.

```
int        i;
tab        lt_tab;
userobject luo_temp;
u_dw       lu_dw

for i = 1 to upperbound( awo_control )
 if( TypeOf( awo_control[i] ) = Tab! ) then
  lt_tab = awo_control[i];
  of_sb_get_dwobjects( lt_tab.control ); // Recursive call
 elseif( TypeOf( awo_control[i] ) = UserObject! ) then
  luo_temp = awo_control[i];
  of_sb_get_dwobjects( luo_temp.control ); // Recursive call
 elseif( TypeOf( awo_control[i] ) = DataWindow! ) then
  // iu_dw is an instance variable that is an array of u_dw datawindow controls
  iu_dw[ upperbound(iu_dw) + 1 ] = awo_control[i];
 end if
next
```

Step 4: In the Timer event for w_sheet, add the following code to check row changes happened to all the DataWindows:

```
int i

for i = 1 to upperbound(iu_dw)
 iu_dw[i].of_sb_verticalscroll()
next
```

Step 5: Add an instance variable for u_dw.

```
long il_old_first_row = -1;
```

Step 6: Add a new function of_sb_verticalscroll( ) for u_dw.

```
long ll_new_firstrow, ll_counter // Check if only one row per page is being
 displayed in dw
ll_new_firstrow = long( this.object.DataWindow.FirstRowOnPage);
if (ll_new_firstrow <> il_old_first_row) then
 il_old_first_row = ll_new_firstrow
 this.trigger event ue_scrollvertical( )
end if;
```

Step 7: Define a user-defined event for u_dw such as ue_scrollvertical to replace ScrollVertical.

Step 8: Place the same code that you plan to put in ScrollVertical in the user-defined event.

6.4.3.2.2 RetrieveRow event

Description

The RetrieveRow event is unsupported.

Workaround

Move the relevant logic to the RetrieveEnd event.

Example

The original script in the RetrieveRow event:

```
if row < 1 then return
// for every single time, check whether the row should be deleted or not.
if f_find(istr_dwnum.dw,istr_dwnum.id
+"='"+this.getitemstring(row,istr_dwnum.id)+"'")>0 then
 this.deleterow(row)
end if
```

The modified RetrieveEnd event (preceding logic is moved to the RetrieveEnd event):

```
long li_row
// loop all rows retrieved from the database and find out which row should be
 deleted.
for li_row = 1 to rowcount
 if f_find(istr_dwnum.dw,istr_dwnum.id+"='"+this.getitemstring(li_row,istr_dwnum.id
 +"'")>0 then
   this.deleterow(li_row)
     li_row --
 end if
next
```

6.4.3.3 Properties

6.4.3.3.1 LiveScroll property

Description

The LiveScroll property for DataWindow control is unsupported.

Workaround

When the LiveScroll property is enabled, it does not take effect on the Web. A row can only be selected by mouse clicking.

6.4.4 Controls in a DataWindow

6.4.4.1 Large Binary/Text database OLE object

Description

The Large Binary/Text database OLE object for DataWindow is unsupported.

Workaround

This workaround only applies to the BitMap OLE object.

If a column with any large Binary/Text Database OLE object is used for displaying graphs, do the following steps to work around this issue:

Step 1: Replace the column with a Graph control.

Step 2: Retrieve the content of the column in the table related with the DataWindow using a SELECTBLOB SQL statement.

Step 3: Call the SetPicture function of this Graph control.

6.4.4.2 Column control

6.4.4.2.1 Char data type column

Description

When you set the data type of a column as *char* with a specified length, the value of the column data type retrieved by Appeon is different from that in PowerBuilder. The value you get in Appeon is *char*, but the value you get in PowerBuilder is *char(n)*.

Workaround

Add one more condition while using the returned value.

Example

The original script:

```
string ls_datatype
ls_datatype = dw_1.object.group_id.coltype
if(ls_datatype = “char(50)”) then
  ...
else
  ...
end if
```

Add one more conditional statement, as shown in the following modified script.

The modified script:

```
string ls_datatype
ls_datatype = dw_1.object.group_id.coltype
if(ls_datatype = “char(50)” or ls_datatype = "char") then
  ...
else
  ...
end if
```

6.4.4.2.2 Evaluating DataWindow expressions in scripts

Description

When using global functions in DataWindow expressions to dynamically change the attributes of DataWindow objects at run time, this method does not work well with an Appeon application, for the DataWindow expressions are only evaluated once.

Workaround

Modify a computed expression on the DataWindow in order to force the expression to re-evaluate.

Note: Generally speaking, DataWindow expressions will slow-down the initial display or subsequent refresh of DataWndows. As such, Appeon recommend you reduce the usage of DataWindow expressions if possible, especially in the following situations:

- Avoid using DataWindow expressions for computing and setting column properties.
- Avoid setting sort and filter criteria directly for a DataWindow object. Instead, write the sort and filter criteria in the SQL statement of the DataWindow object. As noted previously, it is faster to use SQL statements than DataWindow functionality.

Example

An expression like this will not re-evaluate itself: Expression: f_color()

After making a change that would cause f_color() to return a different value (i.e. selecting a different preferred color from a drop-down), the application has to slightly change the size of the DataWindow in order to force the expression to re-evaluate.

The modified script:

```
integer li_dw_width, li_dw_height
li_dw_width=dw_1.width
li_dw_height=dw_1.height

dw_1.width=li_dw_width-4         // Squeeze the DW
dw_1.height=li_dw_height-4

dw_1.width=li_dw_width         // Restore to original size
dw_1.height=li_dw_heigh
```

6.4.5 Functions of DataStore object

Description

CategoryCount,Clipboard, CopyRTF, DataCount for DataStore, FindCategory, FindSeries, GetBorderStyle, GetData, GetDataPieExplode, GetDataStyle, GetDataValue, GetSeriesStyle, PrintCancel, ResetDataColors, SeriesCount, SeriesName, SetDataPieExplode, SetDataStyle, SetDetailHeight, SetSeriesStyle.

Workaround

Replace the DataStore with a DataWindow control and call the corresponding functions of the DataWindow control.

6.5 DBParm parameters in Database

6.5.1 ConnectString parameter

Description

The ConnecString parameter is unsupported.

Workaround

Connect to the database via JDBC, and specify the Data Source name in AEM or dynamically set up the database connection. For more details, please refer to the Chapter 4, *Database Connection Setup* in *PowerServer Configuration Guide for .NET* or PowerServer Configuration Guide for J2EE.

7 FAQ & Workarounds

This chapter lists some frequently asked questions and workaround tips regarding the Appeon application architecture or product features.

7.1 What files will go to the plugin folder and how to access them

What files will go to the "plugin" folder?

For each Appeon application, a "plugin" folder will be created automatically under the application directory (so called sandbox on the mobile device and so called cache directory on the Web client) to store any external files created or used by the application. These external files include

- Any files created by the API calls, such as photos or videos created by the Camera APIs, bar code images created by the Barcode APIs, log files created by the of_log API etc.
- Any files downloaded from the Appeon File Server, such as PDF files, image files, multimedia files, text files, etc.

  For instructions on how to upload and download files from Appeon File Server, see Configuring and deploying Appeon File Server and Uploading and downloading files.
- Any files deployed to PowerServer from PowerServer Toolkit and downloaded from the PowerServer at runtime, such as PDF files, image files, multimedia files, text files, DLL/OCX files, etc.

  For instructions on how to deploy these files to PowerServer, see Deploy External Files in *PowerServer Toolkit User Guide*.
- Any files created by the SaveAs function of the DataWindow control, the DataWindowChild object, or the DataStore object, such as PDF files, text files, HTML files, excel files etc.

How files go there?

All of the above files are automatically saved to the "plugin" folder, no matter they are new files created by the API or function calls or pre-existing files downloaded from the server to the client.

You can also save a file to the blob fields in the database, then read the blob data via the SQL statement, and then write the file to the "plugin" folder by using the PB FileWrite function.

How to reference these files in the script?

To programmatically use the files under the "plugin" folder, you will need to take advantage of a function provided by Appeon, to get the full directory of the application directory. You can either use the AppeonGetCacheDir global function or the of_getcachedir function of the AppeonExtFuncs object. They work exactly the same. For example, appeongetcachedir() + "/plugin/test.pdf" or of_getcachedir() + "/plugin/test.pdf".

For some functions, such as the PB file functions (except for operating the INI files), the directory is automatically default to the "plugin" directory. For example, FileExists ("c:\test \test.txt") will be executed as FileExists (appeongetcachedir() + "/plugin/test.txt").

7.2 How to generate a PDF file and send it via email

Follow steps below to generate a PDF file and send it via email on a mobile device:

Step 1: Generate the PDF file.

- Method 1: Generate the PDF file by calling the SaveAs function of the DataWindow control, the DataWindowChild object, or the DataStore object.

  Syntax:

  ```
  saveas(filename, PDF!, colheading{, encoding})
  ```

  For example:

  ```
  dw_1.saveas("test.pdf", PDF!, True)
  ```

  Note:

  1. filename cannot be empty string or null value.
  2. The file path will be ignored, because the generated file is automatically saved to the %cachedir%/plugin/ directory (%cachedir% indicates the cache directory for the current mobile application). You can determine the cache directory by calling the AppeonGetCacheDir function, as shown in the code example below for step 2.

- Method 2: Download an existing PDF file from the Appeon File Server to the mobile device.

  The PDF file will be automatically downloaded to the %cachedir%/plugin/ directory. You can determine the cache directory by calling the AppeonGetCacheDir function, as shown in the code example below for step 2.

  You can follow the instructions in Configuring and deploying Appeon File Server to install and configure the Appeon File Server and Uploading and downloading files to upload and download the PDF file from the Appeon File Server.

  Syntax:

  ```
  appeonfileservice lnv_ftp
  lnv_ftp = create appeonfileservice

  string ls_server
  integer li_port
  string ls_connectstring

  ...
  ls_connectstring = "username=appeon;password=appeon"

  ...
  ll_re = lnv_ftp.of_appeondownload ("test.pdf","") // download pdf file

  ...
  ```

Step 2: Send the PDF file.

- Call the of_sendmail function of eon_mobile_awsex object to send the PDF file as attachment.

Syntax:

```
eon_mobile_awsex lnv_aws

eon_mobile_str_mailcontent lstr_content
string ls_recipient[]
string ls_cc[]
string ls_bcc[]
string ls_attachmentfile[]
integer li_re
...
lnv_aws = create eon_mobile_awsex
lstr_content.s_subject = "test mail"
lstr_content.s_notetext = "This is a test mail"
ls_recipient[1] = "support@appeon.com"
ls_attachmentfile[1] = appeongetcachedir() + "/plugin/test.pdf"
...
li_re = lnv_aws.of_sendmail (lstr_content, ls_recipient, ls_cc, ls_bcc,
 ls_attachmentfile)
```

Note: You should have an email account configured already on the mobile device to send the PDF file via email as attachment.

7.3 How to make a phone call in the mobile application

Although Appeon provides no APIs to call a phone number within the mobile application, you can still implement it using the following code (effective for both iOS and Android applications):

```
inet l_inet
string ls_code, ls_protocol
int rc

ls_code = "800-555-1212"
ls_protocol= "tel:"

rc = messagebox (ls_code~r~n, 'Dial it?', question!, yesno!, 1)

if rc = 1 then
   l_inet = create inet
   l_inet.hyperlinktourl ( ls_protocol+ls_code )
   Destroy l_inet
end if
```

7.4 How to open Google Map in the mobile application

You can call the Google Map native app or the Google Map Web app, depending on the mobile OS. For iOS, you can either call the native app or the Web app; while for Android, you can only call the Web app.

To call the Google Map native app:

First define a function to call the Google Map native app via its URL scheme. Note that URL scheme is supported in iOS only.

```
public function integer of_gm_direction (string as_saddr, string as_daddr , string
 as_dirmode);

// Only iOS can support google map URL Scheme
// as_dirmode:driving,transit,bicycling,walking
```

```
string ls_url, ls_urlscheme
inet l_inet
integer li_re

ls_urlscheme = "comgooglemaps://?saddr="+as_saddr+"&daddr="+as_daddr
+"&dirflag="+as_dirmode

l_inet = create inet
li_re = l_inet.hyperlinktourl(ls_urlscheme )

destroy l_inet
end function
```

Call the function by passing in the start address, destination address, and direction mode. For example:

- The address can either be the name of places (using + as separator) or the latitude and longitude.
- The direction mode can be: driving, transit, bicycling, walking.

```
of_gm_direction("22.549721,113.948371","22.5584634,114.1337181","bicycling")
```

Or

```
of_gm_direction("appeon+sz+Guangdong+china","22.5584634,114.1337181","driving")
```

To call the Google Map Web app:

First define a function to call the Google Map Web app via URL and parameters.

```
public function integer of_gmweb_direction (string as_saddr, string as_daddr ,
 string as_dirmode);

// as_dirmode:d,t,b,w

string ls_url, ls_as_dirmode4web
inet l_inet
integer li_re

ls_url = "https://maps.google.com/maps?saddr="+as_saddr+"&daddr="+as_daddr
+"&dirflag="+ls_as_dirmode4web

l_inet = create inet
li_re = l_inet.hyperlinktourl(ls_url )
Destroy l_inet
return li_re
end function
```

Call the function by passing in the start address, destination address, and direction mode. For example,

- The address can either be the name of places (using + as separator) or the latitude and longitude.
- The direction mode can be: d (for driving), t (for transit), b (for bicycling), w (for walking).

```
of_ gmweb _direction("22.549721,113.948371","22.5584634,114.1337181","b")
```

Or

```
of_ gmweb _direction("appeon+sz+Guangdong+china","22.5584634,114.1337181","d")
```

7.5 How to download an image to the mobile device and show it in the application

Follow steps below to download an image file to the mobile device and show the picture in the mobile application:

Step 1: Install the Appeon File Server. Detailed instructions can be found in Configuring and deploying Appeon File Server.

Step 2: Upload the image file to the Appeon File Server, and then download it from the Appeon File Server to the mobile device. Here are the specific steps:

1. Add Appeon Workarounds PBL to your PowerBuilder application.
2. Define a variable and create an instance for the AppeonFileService object.
3. Call of_logonfileserver to connect to the file server.
4. Call of_appeonupload to upload the image file to the file server.
5. Call of_appeondownload to download the image file from the file server to the mobile device.

   The file is automatically saved to the %cachedir%/plugin/ directory. %cachedir% indicates the cache directory for the current mobile application. You can determine the cache directory by calling the AppeonGetCacheDir function.
6. Call of_logofffileserver to disconnect from the file server.

Step 3: Access the image file through AppeonGetCacheDir()+"/plugin/"+%filename% in your mobile application.

Alternatively, if you only need to show the image file in the application and does not need to download the image to the mobile device, you can also take advantage of an Appeon enhanced feature to efficiently implement it. Appeon has enhanced the PictureName property of the Picture/PictureButton/PictureHyperLink control, the DataWindow column's content (when Display as Picture option is on), and the DataWindow Bitmap function to support the Internet URL. Therefore, you can post the image file to a Web site that is accessible by the mobile device, and utilize the aforementioned features to directly show the picture in the mobile application.

7.6 How to replace the icon that shows in the IE tab or address bar

Description

When you launch an application in Internet Explorer, an icon shows in the IE tab and the address bar. By default, the icon is the IE icon, but you can change it to any other icon.

Workaround

Step 1: Prepare an icon that you want to display in the IE tab and the address bar. Name it to "app.ico".

Step 2: Go to the Web folder where the application is deployed, for example, C:\Inetpub \wwwroot\codeexamples, copy and paste the **app.ico** file to the **images** folder, and modify the **x32_application.htm** and **x64_application.htm** files by following instructions below:

The original code:

```
<head>
 <title>Appeon Web Library</title>
</head>
```

Should be added with the following line of script (in bold text):

```
<head>
 <link rel="Shortcut Icon" href="./images/app.ico">
 <title>Appeon Web Library</title>
</head>
```

7.7 How to replace Appeon image that displays at the running of applications

Description

When you launch an application in Internet Explorer, an image () shows in the browser for a moment, indicating the start of the loading process. The image is pre-defined in Appeon, but you can change it to any other image.

Workaround

Once you change the Appeon-defined image to your own image, all applications deployed to the server will show the new image at the beginning of application-loading process.

Step 1: Prepare an image that you want to display at the beginning of application-loading process. Name it to "awl_loading.gif".

Step 2: Go to the folder where the Appeon-defined image is stored, and replace the old awl_loading.gif with the new awl_loading.gif. You need to replace awl_loading.gif in the following two folders: *%powerserver%\weblibrary_ax\debug\image* and *%powerserver% \weblibrary_ax\release\image*.

7.8 How to remove the Internet Explorer menu

Description

When a Web application is opened in Internet Explorer, you may see both the Internet Explorer menu and the Web application menu. It is possible to design an HTML file or C++ program to remove the Internet Explorer menu.

Workaround

Method 1: In the application folder under the Web server Web root, create an HTML file named appeon.html for loading the Web application. For example,

```
<html>
 <head>
    <title>Appeon Web Library </title>
 </head>
 <script language="javascript">
    function startApp() {
    g_newWindow = window.open("index.html", "_blank",
```

```
    "location=no,titlebar=no,toolbar=no,menubar=no,status=no,resizable=yes",false);
}
 </script>
 <body>
  <script language="javascript">
    startApp()
  </script>
 </body>
</html>
```

Note: In order for sFeatures settings to take effect, the sName argument in the Open method must be "_blank". This opens a new Internet Explorer window for index.html (the index page of the Web application) and hides the Internet Explorer menu.

Instead of loading Index.html as the entry page for Appeon Web applications, load the appeon.html file using a URL similar to this: http://host:port/appname/appeon.html. When the Open method in the appeon.html file is triggered, the Index.html page will be loaded in a new Internet Explorer window.

Method 2: Create a C++ program that utilizes COM API on the Client side.

The sample code in the C++ program is as follows:

```
// Start a new Internet Explorer as a separate process
 IWebBrowser2* pIE = NULL;
 HRESULT hr;
 hr = CoCreateInstance(CLSID_InternetExplorer, NULL, CLSCTX_SERVER,
    IID_IWebBrowser2, (LPVOID*)&pIE); // if open IE OK
  if (SUCCEEDED(hr)) {
    pIE->put_Visible(TRUE);
    pIE->put_AddressBar(FALSE);
    pIE->put_MenuBar(FALSE);
    pIE->put_StatusBar(TRUE);
    pIE->put_ToolBar(FALSE);
    pIE->put_FullScreen(FALSE);
    COleVariant vtEmpty;
    CString strURL = "http://appeonserver:81"; // The URL to be opened by the
 program.
    BSTR bstrURL = strURL.AllocSysString();
    pIE->Navigate(bstrURL, &vtEmpty, &vtEmpty, &vtEmpty, &vtEmpty);
    ::SysFreeString(bstrURL);}
```

If the user runs the C++ program on the Client machine, the Web application will be opened in an Internet Explorer browser and the display mode of the Internet Explorer browser is specified in the C++ program.

7.9 How to deploy an Appeon application without PowerServer Toolkit

Description

PowerServer Toolkit is needed to deploy the application to at least one PowerServer and Web Server. Once deployed to a PowerServer and a Web Server, you can replicate the deployed application to other PowerServer and Web Servers without using PowerServer Toolkit.

Workaround

Use the Package Wizard provided in the PowerServer Toolkit to generate a portable installation package for your Appeon deployed application and install the package to the other PowerServer and Web Servers. For step-by-step instructions on how to package and

install applications, please refer to Chapter 10, *Packaging Applications* in *PowerServer Toolkit User Guide*.

7.10 How to log in the Appeon Web application with single sign-on

Suppose you have an Appeon Web application and another Web application (created using Visual Studio, COBOL, Web Services, SOA, etc.); and you want to access the Appeon Web application from another Web application (suppose called ABC) without needing to log in again since you have already logged into the application ABC. In such a scenario, you can pass the user credentials to the Appeon Web application without requiring re-login using either of the following two methods.

Method 1: Apply the command line argument.

Appeon supports the CommandParm function and the command line argument in the Open event of a PowerBuilder application. These features can be applied for implementing single sign-on.

The command line argument can be passed to an Appeon application in the following way:

http://192.0.1.94:8080/MyTest/index.html?user=appeon&password=appeon

This attaches the string "*index.html?arguments*" to the end of the original application URL ("*index.html*" must be included in the string).

If the user wants to launch an Appeon application after logging on to another application (created using Visual Studio, COBOL, Web Services, SOA, etc.), the application passes the user information via the URL of the Appeon application, and the user starts the Appeon application without further login procedures.

It is also possible to pass the session ID only. A table is created in the database for keeping the session information of the application, with a session ID assigned to each session, and the session information containing user information. When the Appeon application is launched with the session ID as its command line argument, the application reads from the database table the user information and authenticates the user. The user can start the Appeon application without further login procedures.

Method 2: Use the single sign-on server to pass user information.

If you have set up the single sign-on server, you can take advantage of the single sign-on server to pass the user credentials. Normally the single sign-on server provides the authentication mechanism interface via Web service/EJB/JSP function etc. Make sure that your PowerBuilder can support the interface, then the Appeon Web application should work with it.

7.11 How to add headers & footers to a Web application

Description

Use this solution to add headers and footers to a corporate website with an Appeon application in a frame.

Workaround

Create a new frame page with a header and footer. Set the URL that is used to access the deployed application to the initial page of the content frame.

Step 1: Prepare two HTML files: *header.html* and *footer.html*.

Step 2: In Microsoft FrontPage, create a new frame page that contains header, footer and content frames. To create a new frame page, click **File** | **New** | **Page**. In the *Frames Page* tab, select *Header, Footer and Contents Template*.

Step 3: Set *header.html* as the initial page of the header frame, and set *footer.html* as the initial page of the footer frame.

Step 4: Set the URL that is used to access the deployed application as the initial page of the content frame.

Step 5: Put the frame page in the same Web root as the original deployed application and provide the frame page name to the user.

Now the user can directly open the frame page which opens the deployed application. Headers and footers have already been added to the deployed application.

7.12 How to get the user name and password of the operating system

To get the user name and password of the operating system, follow the two steps:

Step 1: Package the GetUserName function into a DLL then call the DLL in PowerBuilder.

Step 2: Deploy the DLL to PowerServer and download the DLL to the client needed. You can get the detail information of the GetUserName function from the following Website: http://msdn.microsoft.com/en-us/library/windows/desktop/ms724432(v=vs.85).aspx.

7.13 How to modify the storage location of Web application files in PowerServer Toolkit

PowerServer Toolkit will generate and store Web application files on the local machine before deploying it to PowerServer. You can configure to store these files to other location rather than the default one. By default the Web files will be stored in %Appeon_Developer% \Project\%application_name% (e.g., *C:\Program Files\Appeon\PowerServer\Toolkit\Project \sales*). %Appeon_Developer% indicates the installation path of PowerServer Toolkit.

How to configure the location for Web files

Step 1: Open the ADTConfig.xml file in the directory Appeon\PowerServer\Toolkit.

Step 2: Find the following lines:

```
<PathCfg>
<Project value="%Appeon%\PowerServer\Toolkit\Project\"/>
</PathCfg>
```

Step 3: Replace the value in bold with another location where you want to store the Web files.

7.14 How to integrate Appeon Web applications with JSP/ASP

7.14.1 Applying Appeon CommandParm and Hyperlink features

Description

If your application needs to pass parameters to a JSP/ASP application, use the following method.

Workaround for passing parameters from a JSP/ASP application to an Appeon application

Apply the JSP/ASP programming method to add parameters to the URL of the Appeon application. Based upon the parameters, any functionality can be built in the Appeon Web application, such as opening windows and retrieving data to provide client-side integration.

The Appeon application receives the parameters using the CommandParm function and CommandLine parameter of the Open event.

Workaround for passing parameters from an Appeon application to a JSP/ASP application

Appeon supports PictureHyperLink and StaticHyperLink window controls. For example, you can statically or dynamically assign the URL of the JSP/ASP Web application to *http://www.x.x/index.asp?aid=x&bid=y&cid=z* in the Clicked event of a PictureHyperLink or StaticHyperLink window control and send the parameter from the Appeon application to a JSP/ASP application.

Appeon also supports the HyperLinkToURL PowerScript function. The developers can also apply this function to pass the parameter to a JSP/ASP application through automatic code rather than being user-initiated. The Web application needs to be refreshed while receiving the parameter.

7.14.2 Using Internet Explorer Frame

Description

Appeon applications can be accessed in an Internet Explorer frame. It is possible to set the Appeon application and the ASP/JSP application in two different frames of the same browser.

Workaround

Take the following steps to build up integration using IE frames:

Step 1: Divide the IE browser into two frames; one for running the Appeon application and one for running the ASP/JSP application.

Step 2: Apply the Appeon CommandParm feature for launching the Appeon application from the ASP/JSP application.

Step 3: Apply the Appeon Hyperlink feature for launching the ASP/JSP application from the Appeon application.

7.14.3 Integration through intermediate n-Tier Server-level solutions

Description

It is possible to pass parameters between Appeon applications and JSP/ASP applications by applying server-level integration. The information that is passed can be stored at any of the tiers in the n-Tier environment, including the Client PC, Application Server, or the Database Server.

Workaround

- The information can be stored and read on the Client PC operating system through the signed and secure Appeon ActiveX.

Both Appeon and JSP/ASP application read and write a normal Client PC operating system file. The JSP/ASP application needs the ability to access the Client PC operating system file through ActiveX or a Plug-in, etc.

See the following example for how to store and read the information on the Client PC operating system DLL. Appeon calls to a Client PC operating system DLL file in the same way as in PowerBuilder using the Appeon ActiveX:

```
Var objForm
  set objForm=Server.CreateObject("Scripting.Dictionary")
  set tStream = Server.CreateObject("adodb.stream")
```

- The intermediate information can be stored in a Database Server table.

  Both applications can read and write a normal RDBMS database table. Information such as orders, products, customers, or loans can be stored in a database table. After the information is stored in the database table, other applications can trigger a user-initiated event or simple automatic timer event to get updated information.

- The intermediate information can be stored in a file on the Application Server.

If both the Appeon and JSP/ASP applications call to the same DLL, the developer can make use of functionality provided by DLLs for setting up communication between the applications.

7.15 No Workarounds

Not every unsupported feature can be worked around or need to be worked around. Some features are small or trivial or may not necessarily cause functionality loss even if they are not modified or worked around.

If the unsupported feature is flagged as "Have to modify" in the Unsupported Feature Analysis report while you cannot find a workaround for it, please send a test case to <support@appeon.com> for help.

Index

F

G

H

I

P