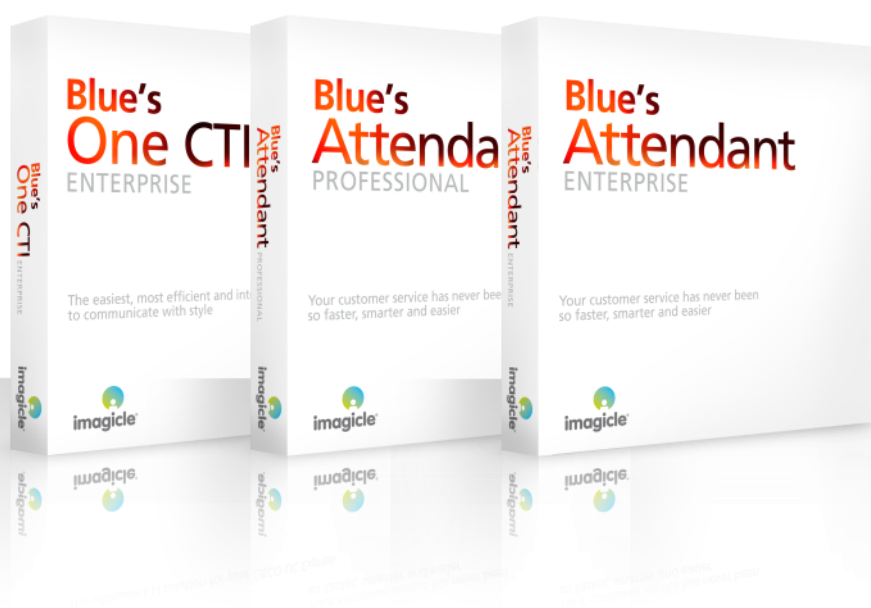




Blue's One CTI Enterprise Blue's Attendant Pro/Enterprise SDK - plugin creating tools

rel. 1.2 ENG 01-06-2012



Plug-in creating introduction

This document has the purpose of explaining the creation of a search plug-in to be used with Blue’s One CTI Enterprise, Blue’s Attendant Professional and Blue’s Attendant Enterprise (herein called Blue’s One CTI - Attendant). The project shall be composed of one or more dll’s, depending on the case. The dll that will implement the ISearch interface will be better explained later and its name will have to follow this form:

Search< plug-in name>.dll

Along with this document, a generic project shall be provided in order to start the creation of your own plug-in. The use of Microsoft .NET Framework 2.0 is required for the development of the plug-in.

The Plug-in dll’s must be saved in the software installation folder under the **SearchPlugIn folder** (usually {program files}\Imagicle Blue's Attendant\SearchPlugIn)

Contents

The structures	3
The plug-in interface	5
Plug-in notification events	8
Option form	8
Resources nationalization	9
User information notices	9
TelcenLog class	9

The structures

The only structure that the main software delivers to the plug-in and vice-versa is the Contact structure which contains information concerning a phonebook contact.

Furthermore, two enumerators are available; `PhoneNumberType` which defines the telephone type, and `LanguageOption` for the plug-in language and nationalization.

The Contact structure has the following form:

```
public struct Contact
{
    public bool bEmptyRecord;
    public Icon m_oIcon;

    public String sContactID;
    public String sIDRubrica;
    public String sContact;
    public String sPhoneNumber;
    public String sCompany;

    public String sMailAddress;
    public String sMailAddress1;
    public String sMailAddress2;

    public String sCustomType;
    public PhoneNumberType oNumType;

    public void InitializeContactStruct()
}
```

The `Contact` structure contains a function that initializes the structure and that has to be invoked each time a `Contact` type object is created.

Variables within the structures are:

- `bool bEmptyRecord`
Flag indicating if the record contains valid or invalid data. By default, it is set to false.
- `Icon m_oIcon`:
contact icon that will be displayed in the contact search list. Its dimensions must be 32x32 at 256 colors.
- `String sIDRubrica`:
unique ID of the created phonebook. A string that contains a maximum of 6 characters and that will be used to distinguish a plug-in from another one.
- `String sContactID`:
ID of a specific contact. This value will be used to open a contact card
- `String sContact`:
contact value, usually Name and Surname. The software will be using it with the same format as provided by the plug-in (empty by default)



- **String** sPhoneNumber:
Contact phone number (empty by default)
- **String** sCompany:
Contact company name (empty by default)
- **String** sMailAddress:
Contact primary e-mail address
- **String** sMailAddress1:
Contact secondary e-mail address
- **String** sMailAddress2:
Contact secondary e-mail address
- **String** sCustomType: customizable phone number type. The standard set includes Office, Fax, Home, Mobile, Other, Telephone. A custom type can also be added in this variable.

Variable **PhoneNumberType** of the structure shall be set as PHN_TYPE_CUSTOM

- **PhoneNumberType** oNumType:
Telephone number type. The available types are defined by the enumerator **PhoneNumberType** below described.
- **public** InitializeContactStruct:
Structure initialization function

PhoneNumberType enumerator:

This enumerator defines a telephone number type.

It may have the following values:

- PHN_TYPE_HOME: telephone number type 'Home'
- PHN_TYPE_BUSINESS: telephone number type 'office'
- PHN_TYPE_MOBILE: telephone number type 'Mobile'
- PHN_TYPE_FAX: telephone number type 'Fax'
- PHN_TYPE_OTHER: telephone number type 'Other'
- PHN_TYPE_GENERIC_PHON: telephone number type 'Phone'
- PHN_TYPE_CUSTOM: custom telephone type. When selected, the telephone type is represented by the value of the sCustomType variable
- PHN_TYPE_SDIALER: telephone number type "

LanguageOption enumerator

This enumerator tells which language shall be used for the plug-in nationalization.

It may have the following values:

- IT: Italian language
- EN: English language
- ES: Spanish language
- FR: French language
- DE: German language



The plug-in interface

The plug-in must implement the [ISearch](#) interface. Its functions are listed below and are used to search contacts in the phonebook, search a number in the phonebook, open the contact card.

The interface functions are:

- [bool](#) IsLoading;
- [bool](#) InitializeComponent([String](#) sFileIni, [String](#) sLogFolder, [bool](#) bEnableLog, [LanguageOption](#) oLanguage);
- [void](#) EnableLog([bool](#) bEnable);
- [void](#) SetPlugInLanguage([LanguageOption](#) oLanguage);
- [bool](#) CloseComponent();
- [void](#) SetMaxResultCount([int](#) iMaxResult);
- [bool](#) ReloadContactData();
- [String](#) GetSearchDescription();
- [Icon](#) GetPhoneBookIcon();
- [Icon](#) GetPhoneBookIconForOption();
- [Control](#) GetOptionControlForm([DelegateSettingsChanged](#) oDelegate, [LanguageOption](#) oLanguage);
- [String](#) GetPhoneBookID();
- [String](#) GetOutgoingPrefix();
- [bool](#) SaveOptionSettings();
- [Contact](#) FindContact([String](#) sPhoneNumero);
- [List<Contact>](#) FindContact([String](#) sContactName, [DelegateProcessContact](#) oHwnWnd);
- [bool](#) OpenContactCard([DelegateContactAdd](#) oDelegateAddContact, [Contact](#) oContact);
- [bool](#) IsEnabled([ref int](#) iNumeroRubricheActive);

Details concerning the use of the functions are listed below.

[bool](#) IsLoading

- **Description:** this property indicates if the plug-in is initializing or not
- **Returns:** [bool](#) indicates the plug-in status
- **Parameters:** None

[bool](#) InitializeComponent([String](#) sFileIni, [String](#) sLogFolder, [bool](#) bEnableLog, [LanguageOption](#) oLanguage)

- **Description:** plug-in initialization function. The application calls it while starting.
- **Returns:** a [bool](#) that indicates the function outcome
- **Parameters:**
 - [String](#) sFileIni: ini file and path where the settings are retrieved and saved
 - [String](#) sLogFolder: folder where the plug-in logs are saved when required
 - [bool](#) bEnableLog: Flag that tells if the log file must be written or not
 - [LanguageOption](#) oLanguage: settings that indicates the language that will be used in the plug-in

[void](#) EnableLog([bool](#) bEnable)

Description: function that enables, according to the provided flag, the creation of a log file

Returns: None

Parameters: [bool](#) bEnable: Flag that sets as enabled or not, the creation of a log file

[void](#) SetPlugInLanguage([LanguageOption](#) oLanguage)

Description: Function that sets the language that will be used within the Plug-in

Returns: None

Parameters: [LanguageOption](#) oLanguage: Language type that will be used



bool CloseComponent()

Description: function is called when Blue's One CTI – Attendant is being closed. The plug-in uses it, when necessary, to close threads that are still open or de-allocate objects allocated during normal operations.

Returns: A **bool** which indicates the function outcome

Parameters: None

void SetMaxResultCount(int iMaxResult)

Description: Function that sets the maximum limit of the records the plug-in can return in correspondence of each search operation. If the function is not set, an internal constant of the plug-in will be used as a limit

Returns: A **bool** which indicates the function outcome

Parameters: **int** iMaxResult: the maximum number of records the plug-in can provide as a result during search operation

bool ReloadContactData()

Description: function that, when required, causes the plug-in to reload or refresh those structures containing a contact list. Returns immediately **true** if not used.

Returns: A **bool** which indicates the function outcome

Parameters: None

String GetSearchDescription()

Description: Function that returns the name of the phonebook the Plug-in connects to. This name will be displayed inside a Tab label, within Blue's One CTI - Attendant options, in the phonebook configuration section

Returns: A **String** containing the name of the phonebook that will be displayed inside a Tab label, in the Blue's One CTI - Attendant options, in the phonebook configuration section

Parameters: None

Icon GetPhoneBookIcon()

Description: function that returns the icon that will be associated to all contacts returned by the plug-in and then displayed within the search lists

Returns: A 32x32 - 256 color **Icon**

Parameters: None

Icon GetPhoneBookIconForOption()

Description: function that returns the icon that will be added as a new section within the Blue's One CTI – Attendant option section.

Returns: A 48x48 **Icon**, **null** if you wish the plug-in to be displayed within the Phonebook section of the Option menu.

Parameters: None

Control GetOptionControlForm(DelegateSettingsChanged oDelegate,LanguageOption oLanguage)

Description: function that returns a form control that will be containing the settings of the created plug-in. the minimum setting that shall be created in the form is a Checkbox that allows to enable or disable the search option via plug-in. The size of the control shall not exceed 350x300. The configuration settings must be saved in a Section of the plug-in and still not included in the ini file provided to the plug-in while initializing

Returns: A **Control** that contains all the settings and values required to the plug-in to operate. The size of the **Control** shall not exceed 350x300.

Parameters:

- **DelegateSettingsChanged** oDelegate: Event that tells the main application that a user applied a modification within the management form of the options.
- **LanguageOption** oLanguage: Language to be used for the nationalization of the plug-in setup form.



String GetPhoneBookID()

Description: function that returns a unique ID of the phonebook. The main application uses this ID to distinguish a plug-in from another one.

Returns: A [String](#) that indicates a unique code representing the phonebook. The string length limit is 5 characters

Parameters: None

String GetOutgoingPrefix()

Description: this function returns the external line code that shall be dialed before the number that will be called. When empty, the default external line code will be used. -1 value indicates that external line codes will not be used in front of the dialed telephone numbers.

Returns: A [String](#) that indicates the external line code to be used in front of the telephone number that will be called.

Parameters: None

bool SaveOptionSettings()

Description: this function is called when the settings of the phonebooks that were modified by the user are saved from the Option menu. The new settings shall be saved within the dedicated section of the ini file, passed to the plug-in at the beginning, through the initialize function.

Returns: A bool that indicates the outcome of the save settings procedure.

Parameters: None

Contact FindContact(String sPhoneNumero)

Description: this feature is called when the software requires to associate an incoming or outgoing call to a contact..

NOTE: the sPhoneNumero passed on variable represents a telephone number that can be displayed in the "+390584123456" format or "00390584123456" format. The Country Code must therefore be managed during a number search in the phonebooks.

Returns: A [Contact](#) structure containing several settings of the contact found in the phonebook. If no contact is found in the phonebook, the bEmptyRecord of the [Contact](#) structure is returned false.

Parameters: [String](#) sPhoneNumero: Searched telephone number

List<Contact> FindContact(String sContactName, DelegateProcessContact oHwnWnd) :

Description: this function searches all contacts within the phonebook that contain the sContactName passed value in one of the name, telephone number, company fields (where present). Each time this event uses a Contact as a parameter, it shall verify if the user has stopped the search operation from the interface. In case the [DelegateProcessContact](#) event call returns [true](#), it would mean search was stopped and the function shall return a partial list of all the contacts found until that moment.

Returns: A [List<Contact>](#). A [Contact](#) structure list containing all the contacts that contain the sContactName value that was passed, in the name, telephone number or company fields.

Parameters:

- [String](#) sContactName: Contact name that will be searched within the phonebook
- [DelegateProcessContact](#) oHwnWnd: event passed by the application and called each time the plug-in finds a matching element. If [true](#) is returned when called, it means the search was stopped and the search shall not proceed further nor return further records, even if already found.

bool OpenContactCard(DelegateContactAdd oDelegateAddContact, Contact oContact)

Description: this function opens the details of the returned contact. The [Contact](#) structure shall contain the ID that will allow you to open the detail card of the passed contact.

Returns: A bool that indicates the outcome of the opening operation of the contact detail card.

Parameters:

- DelegateContactAdd oDelegateAddContact: event called after having added a new contact (NOT used)
- Contact oContact: the Contact structure that contains several settings concerning the contact of which the card will be opened



bool IsEnabled(ref int iNumeroRubricheAttive)

Description: this function is used to monitor the enabling on the search plug-in. As further information, it returns the total number of phonebooks that are active within the plug-in. This, because a plug-in may integrate several phonebooks.

Returns: the function outcome tells if the phonebook is or isn't active. As a reference, it returns the total number of active phonebooks; this, because a plug-in may contain several search facilities in different phonebooks.

Parameters: ref int iNumeroRubricheAttive: total number of active phonebooks returned by the function.

Plug-in notification events

The plug-in contains three delegates to for the management of the events that need to be notified to the main application. Only two of such events are managed.

The events are:

- public delegate bool DelegateProcessContact(Contact oContactFound)
This event is called when the plug-in finds a contact to be returned during a contact search process. This is used to stop, from the main software, a search operation. Returns true if the contact search procedure is requested to stop.
- public delegate void DelegateSettingsChanged()
this event is called the user modifies values from the setting interface within the plug-in. This event is used to notify a change to the main application.
- public delegate void DelegateContactAdd(Contact oNewContactAdd);
Currently unavailable for the integration with new plug-ins.

Option form

The search plug-in will contain a form to display the options required for a correct operation and that the user will be able to modify from the Option menu in the phonebook section.

The implemented form will be of [UserControl](#) type and its size shall not be over 350x300. The nationalization of the form is managed by the plug-in and will be using the language notified by Blue's One CTI - Attendant during the [GetOptionControlForm](#) function.

Each modification to the settings the user will perform within the form must be followed by a call to the [DelegateSettingsChanged](#) message, a delegate provided to the plug-in by the [GetOptionControlForm](#) call.

The plug-in option form must contain at least one Checkbox that will allow the user to enable or disable search through the plug-in.

The graphic form will have to be in the style of the following example:

Outlook

☒ Enable search

NOTE: compatible with Outlook 2000, 2003, 2007 or newest



Resources nationalization

The plug-in is fully responsible for the nationalization of the resources; both regarding the option form and user warnings. Resources will be contained inside EmbeddedResource.resx within the plug-in and already provided by the structure project included within the documentation. The following resource ID's scheme is recommended for the nationalization:

```
ID_LBL_RISORSA_IT  
ID_LBL_RISORSA_EN  
ID_LBL_RISORSA_FR  
ID_LBL_RISORSA_DE  
ID_LBL_RISORSA_ES
```

Therefore, using the information on the language provided by Blue's One CTI – Attendant during the Plug-in initialization will allow the selection of the correct nationalized resource, in accordance with the language.

User information notices

The plug-in may contain information notices that may be displayed to the user but that shall absolutely not be interrupting the application use. The use of a MessageBox is therefore not allowed for any kind of warning.

In order to notify the user in regards of plug-in malfunctions or about any other notification which is considered necessary, a Baloon ToolTip is the only allowed tool. The notification will be displayed through this tool.

TelcenLog class

The structure of each single event saved within the log file shall apply to the following structure:

```
|<DATE> <TIME> | <LOG TYPE> | <CLASS> | <FUNCTION> | <MESSAGE> | <ERRORE CODE>|
```

Where LOG TYPE may use the following values: INFO, DEBUG, WARNING, ERROR, FATAL, GENERIC.

The log file name is chosen by the plug-in creator but it must have a .log extension.

The folder where the log file is saved and the write permission to the file are information that will be provided during the plug-in initialization through the InitializeComponent function.

The project included with the document contains a class which handles logging with the above specified method.

The typical use of the class is as follows:

```
//object intialization  
TelcenLog m_oLog = new TelcenLog();  
m_oLog.EnableLog(true);  
m_oLog.SetClassName("Class Name");  
m_oLog.SetLogFile(@"c:\FileLog.log");  
m_oLog.SetLogMode(LogType.DEBUG_LOG);  
  
//log message being written  
m_oLog.Log(LogType.DEBUG_LOG, "Function Name", "Message to log");  
  
//exception log is being written  
m_oLog.LogException(ex);
```