Creating a PGSuper Extension Agent

|  |  |  |  |
| --- | --- | --- | --- |
| Written By: | RAB | Creation Date: | 04/28/2010 |
| Reviewed By: |  | Last Revised: | 04/28/2010 |
| Project: | PGSuper | WI # | <<#>> |
|  |  | Version | 1.0 - Draft |

# Purpose

The purpose of this work instruction is to describe how to create a new extension agent for the PGSuper application. An extension agent is exactly the same as a regular PGSuper agent, except it can be optionally disabled and is not required for PGSuper to execute property. Extension agents are typically created by 3rd party developers to extend the functionality of PGSuper. WSDOT and TxDOT also have PGSuper Extension Agents that provided optional WSDOT and TxDOT functionality.

# Procedure

The basic procedure for creating an extension agent is basically the same as for creating a regular PGSuper agent. See Creating a PGSuper Agent for details. The differences are highlighted here.

## PGSuper Agent component category registration

1. **Deviation** - Your agent must register itself as a member of the PGSuperExtensionAgent component category. In the Agent.cpp file, replace the generated DllRegisterServer and DllUnregisterServer functions with the following:  
     
   /////////////////////////////////////////////////////////////////////////////  
   // DllRegisterServer - Adds entries to the system registry  
   STDAPI DllRegisterServer(void)  
   {  
    // registers object, typelib and all interfaces in typelib  
    HRESULT hr = \_Module.RegisterServer(FALSE);  
    if ( FAILED(hr) )  
    return hr;  
     
    return sysComCatMgr::RegWithCategory(CLSID\_MyExtensionAgent,  
    **CATID\_PGSuperExtensionAgent**,true);  
   }  
   /////////////////////////////////////////////////////////////////////////////  
   // DllUnregisterServer - Removes entries from the system registry  
   STDAPI DllUnregisterServer(void)  
   {  
    sysComCatMgr::RegWithCategory(CLSID\_MyExtensionAgent,  
    **CATID\_PGSuperExtensionAgent**,false);  
     
    \_Module.UnregisterServer();  
    return S\_OK;  
   }

# User Interface Integration

An extension agent must integrate itself into PGSuper’s user interface in order for its unique functionality to be accessed by the user. User interface integration can be as simple as defining a new report (reports automatically integrate into the report menus and dialogs) or more complex including adding toolbars, menus, accelerator keys, and views. Your extension agent must implement the IAgentUIIntegration interface to enable user interface integration.

## Implementing the IAgentUIIntegration interface

1. Add IAgentUIIntegration to the class declaration of your agent
2. Add COM\_INTERFACE\_ENTRY(IAgentUIIntegration) to the BEGIN\_COM\_MAP()/END\_COM\_MAP() interface map table.
3. Add method definitions to the class definition

// IAgentUIIntegration

public:

STDMETHOD(IntegrateWithUI)(BOOL bIntegrate);

1. Implement the IAgentUIIntegration methods. If the bIntegrate parameter is true, extension agents are integrating with the user interface. If the bIntegrate parameter is false, extension agents must remove themselves from the user interface.  
   ////////////////////////////////////////////////////////////////////  
   // IAgentUIIntegration  
   STDMETHODIMP CExampleExtensionAgent::IntegrateWithUI(BOOL bIntegrate)  
   {  
    if ( bIntegrate )  
    {  
    // Integrate with the user interface  
    CreateMenus();  
    CreateToolBar();  
    RegisterViews();  
    RegisterReports();  
    }  
    else  
    {  
    // Be a good extension agent and remove UI modifiations  
    RemoveMenus();  
    RemoveToolBar();  
    UnregisterViews();  
    }  
     
    return S\_OK;  
   }

## Integration with Menus

Extension agents can add commands to menus and add entirely new menus.

# Adding Reports to PGSuper

An extension agent can add new reports to PGSuper. These reports can contain information obtained from the interfaces published by PGSuper, by your extension agent, by other extension agents, or other outside sources.