[MS-SPPTC]:
User Code Execution Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.

- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.

- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.

- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.

- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.

- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
07/13/2009	0.1	Major	Initial Availability
08/28/2009	0.2	Editorial	Revised and edited the technical content
11/06/2009	0.3	Editorial	Revised and edited the technical content
02/19/2010	1.0	Major	Updated and revised the technical content
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Editorial	Revised and edited the technical content
06/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.

Release: Sunday, December 19, 2010

Table of Contents

1   Introduction

This document specifies the User Code Execution Protocol, which allows a protocol client to call an execution service on a protocol server to run code.

1.1   Glossary

The following terms are defined in [MS-GLOS]:

language code identifier (LCID)
public key
server-activated object (SAO)
XML

The following terms are defined in [MS-OFCGLOS]:

absolute URL
assembly
assembly name
cascading style sheet (CSS)
checked out
CSS
current user
display name
document library
event
event receiver
farm
feature
feature identifier
feature scope
field
front-end Web server
full URL
fully qualified class name
hash
HTML (HyperText Markup Language)
list
list identifier
list item
list item attachment
list item identifier
list server template
login name
managed code
personal view
server-relative URL
shared view
site
site collection
site collection identifier
site identifier
site solution
strong name
URL (Uniform Resource Locator)

user code
user identifier
Web application
Web Part
Web Part Page
Web Part property
Web Part type identifier
Web Part zone
Web Part zone identifier
Web Part zone index
workflow
workflow association
workflow identifier
workflow instance
workflow task

The following terms are specific to this document:

assembly group: A collection of one or more files that contains user code. Each assembly group has a unique name.

feature receiver: A server-side code routine that is called when a feature is activated, deactivated, installed, uninstalled, or upgraded on a computer, server farm, or server cluster.

Web Part chrome: A set of common user interface elements that frame a Web Part within a given zone. The Web Part chrome includes a border, a title bar, and the icons, title text, and verbs menu that appear within the title bar.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2   References

1.2.1   Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[ECMA-262] ECMA international, "ECMAScript Language Specification" ECMA-262, December 1999, http://www.ecma-international.org/publications/standards/Ecma-262.htm

[IEEE754] Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE 754-1985, October 1985, http://ieeexplore.ieee.org/servlet/opac?punumber=2355

[MS-NRBF] Microsoft Corporation, ".NET Remoting: Binary Format Data Structure", July 2007.

[MS-NRTP] Microsoft Corporation, ".NET Remoting: Core Protocol Specification", September 2007.

[MS-WPPS] Microsoft Corporation, "Web Part Pages Web Service Protocol Specification", June 2008.

[MS-WSSFO] Microsoft Corporation, "Windows SharePoint Services (WSS): File Operations Database Communications Protocol Specification", April 2008.

[MS-WSSPROG] Microsoft Corporation, "Windows SharePoint Services: Content Database Programmability Extensions Communications Protocol Specification", June 2008.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

1.2.2 Informative References

[MSDN-SharePointSDK] Microsoft Corporation, "SharePoint Products and Technologies SDK: 2010 API Reference (Technical Preview)", July 2009, http://msdn.microsoft.com/en-us/library/ee557253(office.14).aspx

[MSDN-SHPTSDK] Microsoft Corporation, "Windows SharePoint Services 3.0 SDK", http://msdn.microsoft.com/en-us/library/ms441339.aspx

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary", June 2008.

1.3 Protocol Overview (Synopsis)

This protocol allows a protocol client to call a protocol server, which runs **user code** remotely. In a typical operation, the protocol client sends an **Execute** (section 3.1.5.1) request to the protocol server. The protocol server responds by running the user code and then returns the results. The protocol client can also send a **Ping** (section 3.1.5.2) request to the protocol server to verify that the protocol server is available to receive **Execute** messages. A typical scenario for using this protocol is to remotely run user code within a server **farm** on one or more **front-end Web servers**, which are primarily dedicated to running user code rather than responding to Web requests. This reduces load on the front-end Web servers in the farm that are responding to Web requests and increases their stability by isolating them from problems caused by the running user code. In this scenario, the protocol client is a front-end Web server responding to Web requests, and the protocol server is a front-end Web server primarily dedicated to running user code.

The following diagram summarizes how this protocol is used:
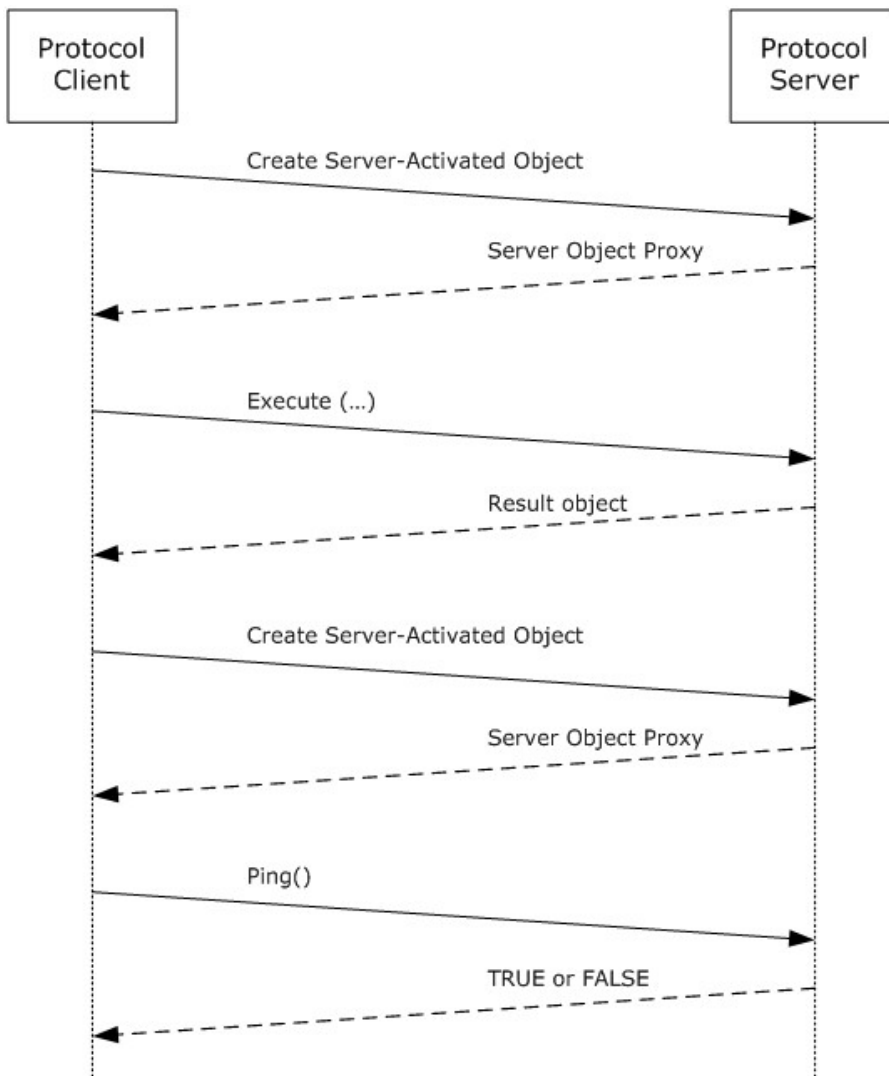
Figure 1: Overview of use of protocol to execute a user code request

1.4   Relationship to Other Protocols

This protocol depends on the .NET Remoting Core Protocol described in [MS-NRTP], and is shown in the following layering diagram:
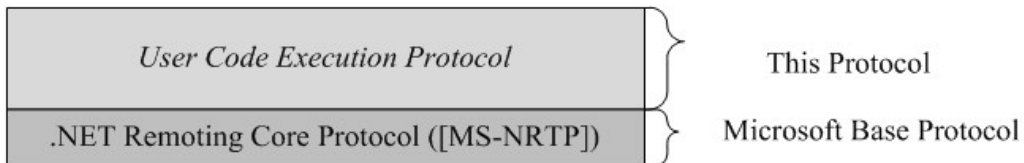
Figure 2: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

The protocol server and protocol client agree on a communication channel as described in [MS-NRTP] section 1.5. The protocol server exposes a **server-activated object**, as described in [MS-NRTP] section 1.3.3, that implements **Execute** (section 3.1.5.1) and **Ping** (section 3.1.5.2) methods.

1.6 Applicability Statement

This protocol applies only to a protocol client calling a protocol server to remotely execute user code on the protocol server. This protocol is intended for use by protocol clients and protocol servers that are both connected by high-bandwidth and low-latency network connections.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

The parameters *userCodeWrapperType* and *executionContex*t of the **Execute** (section 3.1.5.1) method, and the return value of the **Execute** (section 3.1.5.1) method are vendor-extensible fields.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

This protocol is composed of one interface, which is based on the protocol specified in [MS-NRTP]. The interface is **ISPUserCodeExecutionHostProxy**, as specified in section 3.1.5.

The server-activated object implementing the **ISPUserCodeExecutionHostProxy** interface MUST be exposed by the protocol server at the following **URL**:

[baseURL]\portNumber

Where [baseUrl] is a URL reachable by the protocol client, and portNumber is the port the protocol client and protocol server use for communicating with each other.

2.2 Common Message Syntax

The following sections list common data types used by this protocol. See [MSDN-SHPTSDK] for information about implementation-specific details of these data types.

2.2.1 Enumerations

This section specifies enumerations passed between the protocol client and the protocol server.

2.2.1.1 EventScope

This enumeration is used to specify the level at which an **event (2)** runs.

```
namespace Microsoft.SharePoint.UserCode
{
    enum EventScope
    {
            Unknown,
            Item,
            List,
            Web
    }
}
```

Unknown: The event scope is unknown.

Item: The event scope is at the **list item** level.

List: The event scope is at the **list (1)** level.

Web: The event scope is at the **site (2)** level.

2.2.1.2 PartChromeState

This enumeration is used to specify whether a **Web Part** and its **Web Part chrome** are in a normal state, or a minimized state.

```
namespace System.Web.UI.WebControls.WebParts
{
    enum PartChromeState
```

```
        {
                Normal,
                Minimized
        }
}
```

Normal: The Web Part and its Web Part chrome are in a normal state.

Minimized: The Web Part and its Web Part chrome are in a minimized state.

2.2.1.3 PartChromeType

This enumeration is used to specify the type of Web Part chrome to render around the Web Part.

```
namespace System.Web.UI.WebControls.WebParts
{
    enum PartChromeType
    {
            Default,
            TitleAndBorder,
            None,
            TitleOnly,
            BorderOnly
    }
}
```

Default: A border setting inherited from the **Web Part zone** containing the Web Part.

TitleAndBorder: A title bar and a border.

None: No border and no title bar.

TitleOnly: A title bar only, without a border.

BorderOnly: A border only, without a title bar.

2.2.1.4 SPEventReceiverStatus

This enumeration is used to specify the result of an **event receiver**.

```
namespace Microsoft.SharePoint
{
    enum SPEventReceiverStatus
    {
        Continue,
        CancelNoError,
        CancelWithError,
        CancelWithRedirectUrl
    }
}
```

Continue: The event (2) succeeded and the request will continue.

CancelNoError: The event (2) failed and the request needs to cancel without an error.

CancelWithError: The event (2) failed and the request needs to cancel with an error.

CancelWithRedirectUrl: The event (2) failed and the request needs to cancel and redirect to another URL.

2.2.1.5 SPEventReceiverType

This enumeration is used to specify the type of an event (2).

```
namespace Microsoft.SharePoint
{
    enum SPEventReceiverType
    {
        InvalidReceiver        = -1,
        ItemAdding             = 1,
        ItemUpdating           = 2,
        ItemDeleting           = 3,
        ItemCheckingIn         = 4,
        ItemCheckingOut        = 5,
        ItemUncheckingOut      = 6,
        ItemAttachmentAdding   = 7,
        ItemAttachmentDeleting = 8,
        ItemFileMoving         = 9,
        FieldAdding            = 101,
        FieldUpdating          = 102,
        FieldDeleting          = 103,
        ListAdding             = 104,
        ListDeleting           = 105,
        SiteDeleting           = 201,
        WebDeleting            = 202,
        WebMoving              = 203,
        WebAdding              = 204,
        WorkflowStarting       = 501,
        ItemAdded              = 10001,
        ItemUpdated            = 10002,
        ItemDeleted            = 10003,
        ItemCheckedIn          = 10004,
        ItemCheckedOut         = 10005,
        ItemUncheckedOut       = 10006,
        ItemAttachmentAdded    = 10007,
        ItemAttachmentDeleted  = 10008,
        ItemFileMoved          = 10009,
        ItemFileConverted      = 10010,
        FieldAdded             = 10101,
        FieldUpdated           = 10102,
        FieldDeleted           = 10103,
        ListAdded              = 10104,
        ListDeleted            = 10105,
        SiteDeleted            = 10201,
        WebDeleted             = 10202,
        WebMoved               = 10203,
        WebProvisioned         = 10204,
        WorkflowStarted        = 10501,
        WorkflowPostponed      = 10502,
        WorkflowCompleted      = 10503,
        EmailReceived          = 20000,
        ContextEvent           = 32766
    }
```

```
     }
```

InvalidReceiver: Specifies an invalid event type.

ItemAdding: Type of event (2) that is raised before a list item is added to a list (1).

ItemUpdating: Type of event (2) that is raised before a list item is updated.

ItemDeleting: Type of event (2) that is raised before a list item is deleted.

ItemCheckingIn: Type of event (2) that is raised before initiating the check in process for a list item.

ItemCheckingOut: Type of event (2) that is raised before a list item is **checked out**.

ItemUncheckingOut: Type of event (2) that is raised before a list item is reverted from checked out.

ItemAttachmentAdding: Type of event (2) that is raised before a **list item attachment** is added.

ItemAttachmentDeleting: Type of event (2) that is raised before a list item attachment is deleted.

ItemFileMoving: Type of event (2) that is raised before a document is moved to a different **document library**.

FieldAdding: Type of event (2) that is raised before a **field (2)** is added to a list (1).

FieldUpdating: Type of event (2) that is raised before a field (2) is updated in a list (1).

FieldDeleting: Type of event (2) that is raised before a field (2) is deleted from a list (1).

ListAdding: Type of event (2) that is raised before a list (1) is created.

ListDeleting: Type of event (2) that is raised before a list (1) is deleted.

SiteDeleting: Type of event (2) that is raised before a **site collection** is deleted.

WebDeleting: Type of event (2) that is raised before a site (2) is deleted.

WebMoving: Type of event (2) that is raised before the URL of a site (2) is changed.

WebAdding: Type of event (2) that is raised before a site (2) is created.

WorkflowStarting: Type of event (2) that is raised when a **workflow (2)** request is made.

ItemAdded: Type of event (2) that is raised after a list item has been added to a list (1).

ItemUpdated: Type of event (2) that is raised after a list item has been updated.

ItemDeleted: Type of event (2) that is raised after a list item has been deleted from a list (1).

ItemCheckedIn: Type of event (2) that is raised after completing the check in process for a list item.

ItemCheckedOut: Type of event (2) that is raised after a list item is checked out.

ItemUncheckedOut: Type of event (2) that is raised after a list item attachment is added.

ItemAttachmentAdded: Type of event (2) that is raised after a list item attachment is added.

ItemAttachmentDeleted: Type of event (2) that is raised after a list item attachment is deleted.

ItemFileMoved: Type of event (2) that is raised after a document is moved to a different document library.

ItemFileConverted: Type of event (2) that is raised after a document transform is done.

FieldAdded: Type of event (2) that is raised after a field (2) is added to a list (1).

FieldUpdated: Type of event (2) that is raised after a field (2) is updated in a list (1).

FieldDeleted: Type of event (2) that is raised after a field (2) is deleted from a list (1).

ListAdded: Type of event (2) that is raised after a list (1) is created.

ListDeleted: Type of event (2) that is raised after a list (1) is deleted.

SiteDeleted: Type of event (2) that is raised after a site collection is deleted.

WebDeleted: Type of event (2) that is raised after a site (2) is deleted.

WebMoved: Type of event (2) that is raised after the URL of a site (2) is changed.

WebProvisioned: Type of event (2) that is raised before a site (2) is provisioned.

WorkflowStarted: Type of event (2) that is raised when a workflow (2) is loaded and ready to run.

WorkflowPostponed: Type of event (2) that is raised when a workflow (2) is throttled and queued.

WorkflowCompleted: Type of event (2) that is raised after a workflow (2) is finished.

EmailReceived: Type of event (2) that is raised after receiving an external e-mail message.

ContextEvent: This value is not a type of event (2) that gets raised; instead it is used to identify workflow (2) event receivers.

2.2.1.6 SPFeatureCallOutOperation

A **site solution** can contain one or more **features**, each feature can implement a **feature receiver**. When a site solution is installed, activated, upgraded, deactivated, or uninstalled on the protocol client each feature receiver in the site solution is remotely executed on the protocol server. This enumeration is used to specify the types of events (2) that the feature receiver can respond to, except for **InvalidOperation**, which is not an event (2) type but an indication of an unknown event (2) type. A feature receiver can respond to none, some, or all of these types of events (2).

```
namespace Microsoft.SharePoint.UserCode
{
    enum SPFeatureCallOutOperation
    {
        FeatureInstalled,
        FeatureUninstalling,
        FeatureActivated,
        FeatureDeactivating,
        FeatureUpgrading,
        InvalidOperation
```

```
        }
    }
```

FeatureInstalled: The feature has been installed.

FeatureUninstalling: The feature is in the process of getting uninstalled.

FeatureActivated: The feature has been activated.

FeatureDeactivating: The feature is in the process of getting deactivated.

FeatureUpgrading: The feature is in the process of getting upgraded.

InvalidOperation: The requested operation is not recognized.

2.2.1.7 SPFeatureScope

This enumeration is used to indicate **feature scope**.

```
namespace Microsoft.SharePoint
{
    enum SPFeatureScope
    {
        InvalidScope = -1,
        Farm = 0,
        WebApplication = 1,
        Site = 2,
        Web = 3
    }
}
```

InvalidScope: Indicates an invalid feature scope value.

Farm: The feature operates at the farm level.

WebApplication: The feature operates at the **Web application (1)** level.

Site: The feature operates at the site collection level.

Web: The feature operates at the site (2) level.

2.2.2 Classes

This section specifies classes passed between the protocol client and protocol server.

2.2.2.1 EventResults

This class contains the results from running an event receiver on the protocol server.

```
namespace Microsoft.SharePoint.UserCode
{
    class EventResults
    {
        SPEventReceiverStatus status;
        Hashtable changedProperties;
```

```
            string errorMessage;
            string redirectUrl;
        }
    }
```

status: An **SPEventReceiverStatus** (section 2.2.1.4) that indicates whether the event receiver succeeded or not.

changedProperties: A **Hashtable** (section 2.2.2.2) from string to string that contains one key/value pair for each field (2) of the list item that was changed during the execution of the event receiver. It MUST be **null** if the **EventScope** (section 2.2.1.1) in the **SPUserCodeEventHandlerExecutionContext.eventScope** field (section 2.2.2.10) for which this **EventResults** is returned, is not **EventScope.Item**.

errorMessage: Contains an error message when status is not **SPEventReceiverStatus.Continue** (section 2.2.1.4). Otherwise, this property MUST be ignored.

redirectUrl: Contains the **URL** to redirect to if the status is **SPEventReceiverStatus.CancelWithRedirectUrl** (section 2.2.1.4). Otherwise, this property MUST be ignored.

2.2.2.2 Hashtable

This class is used to contain a collection of key-value pairs.

```
namespace System.Collections
{
    class Hashtable
    {
        Single LoadFactor;
        Int32 Version;
        System.Collections.IComparer Comparer;
        System.Collections.IHashCodeProvider HashCodeProvider;
        Int32 HashSize;
        System.Object[] Keys;
        System.Object[] Values;
    }
}
```

LoadFactor: The maximum ratio of elements to number of containers to hold the elements.

Version: The version number of the HashTable contents.

Comparer: Reserved. The value of this field MUST be **null**.

HashCodeProvider: Reserved. The value of this field MUST be **null**.

HashSize: The number of containers that hold the elements in the hash table.

Keys: An array of keys.

Values: An array of values.

2.2.2.3   SPEditorChromeEditorPartSettings

This class contains settings for an editor Web Part, which is a Web Part specifically designed to display and modify properties of the standard Web Part that it is associated with.

```
namespace Microsoft.SharePoint.WebPartPages
{
    class SPEditorChromeEditorPartSettings
    {
        string m_title;
        string m_clientID;
        PartChromeType m_chromeType;
        bool m_useDefaultStyles;
        bool m_allowMinimize;
        PartChromeState m_chromeState;
        string m_renderHtml;
    }
}
```

m_title: Title to be displayed within the **HTML (HyperText Markup Language)** of the Web Part chrome that surrounds the **m_renderHtml**.

m_clientID: HTML identifier of the Web Part chrome HTML.

m_chromeType: A **PartChromeType** (section 2.2.1.3) that specifies the type of Web Part chrome to render around the Web Part.

m_useDefaultStyles: This is **true** if the **cascading style sheet (CSS)** style is reset by the Web Part chrome HTML to match the HTML (HyperText Markup Language) Body element, **false** otherwise.

m_allowMinimize: This is **true** if the **m_renderHtml** can be used within Web Part chrome HTML that has the cascading style sheet (CSS) display value set to none, **false** otherwise.

m_chromeState: A **PartChromeState** (section 2.2.1.2) specifying whether a Web Part and its Web Part chrome are in a normal state, or a minimized state.

m_renderHtml: HTML of the editor Web Part that is displayed within the Web Part chrome HTML.

2.2.2.4   SPEventPropertiesBase

This is an abstract class that provides the event (2) properties that are common to all event (2) types.

```
namespace Microsoft.SharePoint
{
    abstract class SPEventPropertiesBase
    {
        Guid m_siteId;
        string m_receiverData;
        SPEventReceiverStatus m_status;
        SPEventReceiverType m_eventType;
    }
}
```

m_siteId: The **site collection identifier** of the site collection in which the event (2) occurred.

m_receiverData: Contains the value of the *proc_InsertEventReceiver @Data* argument (see [MS-WSSPROG] section 3.1.4.51) that was provided when the event receiver was provisioned.

m_status: An **SPEventReceiverStatus** (section 2.2.1.4), the protocol client MUST set this to **SPEventReceiverStatus.CancelWithError** (section 2.2.1.4). The **SPEventReceiverStatus** (section 2.2.1.4) of the event receiver**'s** execution will be returned from the protocol server to the protocol client in **EventResults** class (section 2.2.2.1).

m_eventType: An **SPEventReceiverType** (section 2.2.1.5) indicating the type of event (2) that occurred.

2.2.2.5   SPException

The **SPException** exception type is a derived class of **System.ApplicationException**, which in turn derives from **System.Exception** as specified in [MS-NRTP] section 2.2.2.7. **System.ApplicationException** implements no additional properties or methods beyond what it inherits from **System.Exception**.

```
namespace Microsoft.SharePoint
{
    class SPException: ApplicationException
    {
        string m_nativeErrorMessage;
        string m_nativeStackTrace;
    }
}
```

m_nativeErrorMessage: Error message if error did not occur in managed code. If the error occurred in managed code this value MUST be **null**.

m_nativeStackTrace: Stack trace if error did not occur in managed code. If the error occurred in managed code this value MUST be **null**.

2.2.2.6   SPItemEventDataCollection

This class contains information about a list item event (2).

```
namespace Microsoft.SharePoint
{
    class SPItemEventDataCollection
    {
        bool m_marshalOnDemand;
        bool m_allowChanges;
        Hashtable m_properties;
        Hashtable m_changedProperties;
    }
}
```

m_marshalOnDemand: This MUST be **false**.

m_allowChanges: This is **true** if the properties in **m_properties** can be changed by the event receiver, **false** otherwise.

m_properties: A **Hashtable** (section 2.2.2.2) from string to string that contains one key/value pair for each property associated with the list item event (2).

m_changedProperties: A **Hashtable** (section 2.2.2.2) from string to string that contains one key/value pair for each field (2) of the list item that was changed during the execution of the event receiver.

2.2.2.7 SPItemEventProperties

This class contains information that an event receiver can use to process a list item scoped event (2).

```
namespace Microsoft.SharePoint
{
    class SPItemEventProperties : SPEventPropertiesBase
    {
        bool m_versionless;
        string m_userDisplayName;
        string m_userLoginName;
        string m_webAbsUrl;
        string m_webRelUrl;
        string m_listTitle;
        Guid m_listId;
        int m_listItemId;
        string m_beforeUrl;
        string m_afterUrl;
        SPItemEventDataCollection m_beforeProperties;
        SPItemEventDataCollection m_afterProperties;
    }
}
```

m_versionless: Indicates if the event (2) was triggered by a versionless change to the list item.

m_userDisplayName: The **display name** of the user whose action initiated this event (2).

m_userLoginName: The **login name** of the user whose action initiated this event (2).

m_webAbsUrl: The URL of the site (2) in which the event (2) occurred.

m_webRelUrl: The **server-relative URL** of the site (2).

m_listTitle: The title of the list (1) in which the event (2) occurred.

m_listId: The **list identifier** of the list (1) in which the event (2) occurred.

m_listItemId: The **list item identifier** of the list item in which the event (2) occurred.

m_beforeUrl: The URL of the list item before the event (2) occurred.

m_afterUrl: The URL of the list item after the event (2) occurred.

m_beforeProperties: An **SPItemEventDataCollection** (section 2.2.2.6) containing the properties of the list item before the event (2) occurred. The event receiver cannot change the value of these properties so **SPItemEventDataCollection.m_allowChanges** (section 2.2.2.6) MUST be **false**.

m_afterProperties: An **SPItemEventDataCollection** (section 2.2.2.6) containing the properties of the list item after the event (2) occurred. The event receiver can change the values of these properties when **SPItemEventPropertiesBase.m_eventType** (section 2.2.2.4) is **SPEentReceiverType.ItemAdding** (section 2.2.1.5) or **SPEentReceiverType.ItemUpdating**

(section 2.2.1.5). **SPItemEventDataCollection.m_allowChanges** (section 2.2.2.6) MUST be **true** in these cases and **false** otherwise.

2.2.2.8 SPListEventProperties

This class contains information that an event receiver can use to process a list (1) scoped event (2).

```
namespace Microsoft.SharePoint
{
    class SPListEventProperties : SPEventPropertiesBase
    {
        string m_userDisplayName;
        string m_userLoginName;
        string m_webUrl;
        Guid m_webId;
        string m_listTitle;
        Guid m_listId;
        string m_fieldXml;
        string m_fieldName;
        int m_templateId;
        Guid m_featureId;
    }
}
```

m_userDisplayName: The display name of the user whose action initiated this event (2).

m_userLoginName: The login name of the user whose action initiated this event (2).

m_webUrl: The URL of the site (2) in which the event (2) occurred.

m_webId: The **site identifier** of the site (2) in which the event (2) occurred.

m_listTitle: The title of the list (1) in which the event (2) occurred.

m_listId: The list identifier of the list (1) in which the event (2) occurred.

m_fieldXml: The **XML** definition of the field (2) that is affected by adding, removing, or updating a field (2) in the list (1) if **SPEventPropertiesBase.eventType** field (section 2.2.2.4) is **SPEventReceiverType.FieldAdding**, **SPEventReceiverType.FieldUpdating**, **SPEventReceiverType**.**FieldDeleting**, **SPEventReceiverType.FieldAdded**, **SPEventReceiverType.FieldUpdated** or **SPEventReceiverType.FieldDeleted** (section 2.2.1.5). Otherwise this MUST be ignored.

m_fieldName: The internal field (2) name that is affected by adding, removing, or updating a field (2) in the list (1) if SPEventPropertiesBase.eventType field (section 2.2.2.4) is **SPEventReceiverType.FieldAdding**, **SPEventReceiverType.FieldUpdating**, **SPEventReceiverType.FieldDeleting**, **SPEventReceiverType.FieldAdded**, **SPEventReceiverType.FieldUpdated** or **SPEventReceiverType.FieldDeleted** (section 2.2.1.5). Otherwise this MUST be ignored.

m_templateId: The **list server template**, as specified in [MS-WSSFO] section 2.2.3.12, that the list (1) is based on.

m_featureId: The **feature identifier** of the feature that created the list (1) or **Guid.Empty** if the list (1) was not created by a feature.

2.2.2.9 SPUserCodeAssemblyGroupId

This class defines a unique identifier for an **assembly group**.

```
namespace Microsoft.SharePoint.Administration
{
    class SPUserCodeAssemblyGroupId : ISerializable
    {
        string groupId;
    }
}
```

groupId: The unique identifier of this assembly group.

2.2.2.10 SPUserCodeEventHandlerExecutionContext

This class contains information that an event receiver uses to run a request.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeEventHandlerExecutionContext : SPUserCodeRemoteExecutionContext
    {
        byte[] eventPropertiesBuffer;
        Guid tranLockId;
        EventScope eventScope;
    }
}
```

eventPropertiesBuffer: The binary serialized format, as specified in [MS-NRBF] section 2.4.3.1, of a subclass of SPEventPropertiesBase (section 2.2.2.4). The type MUST correspond with the value of the **eventScope** field, as specified in the following table.

Value of eventScope	Type of binary serialized object in eventPropertiesBuffer
EventScope.Item	**SPItemEventProperties** (section 2.2.2.7)
EventScope.List	**SPListEventProperties** (section 2.2.2.8)
EventScope.Web	**SPWebEventProperties** (section 2.2.2.29)

tranLockId: This MUST be ignored by the protocol server.

eventScope: An **EventScope** (section 2.2.1.1) enumeration, indicating the scope of this event (2).

2.2.2.11 SPUserCodeExecutionContext

This class contains the context specific data regarding a specific call to run user code on the protocol server.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeExecutionContext
    {
        Guid solutionId;
```

```
        string solutionHash;
        string solutionValidatorsHash;
        string solutionValidationErrorUrl;
        string solutionValidationErrorMessage;
        string proxyAssemblyName;
        string proxyTypeName;
        double? resourceValue;
        bool? resourceQuotaExceeded;
        string currentCulture;
        string currentUICulture;
    }
}
```

solutionId: The unique identifier of the site solution containing user code.

solutionHash: An implementation-specific **hash** that uniquely identifies the content of the site solution containing user code. This value is determined by the implementation of the protocol server.

solutionValidatorsHash: An implementation-specific hash that uniquely identifies the code used to validate the site solution containing user code. This value is determined by the implementation of the protocol server.

solutionValidationErrorUrl: This value represents a URL to redirect to when the most recent attempt to validate the solution has failed. This value MUST be **null** when the most recent attempt to validate the solution has succeeded.

solutionValidationErrorMessage: This value is an error message to render when the most recent attempt to validate the solution has failed and **solutionValidationErrorUrl** is **null**. This value MUST be **null** when the most recent attempt to validate the solution has succeeded.

proxyAssemblyName: The **strong name** of the **assembly** that contains the type specified by the **proxyTypeName** field.

proxyTypeName: The **fully qualified class name** of a type, contained in the assembly specified in the **proxyAssemblyName** field, that the protocol server MUST allow to run at a higher security level.

resourceValue: The highest weighted resource usage value for this site solution divided by the number of invocations of this site solution. For example, if this site solution had run 20 times and consumed a total of 10 units of processing time and 3 units of memory, this value would be 10/20 or 0.5.

resourceQuotaExceeded: This is **true** if the site collection containing this site solution with user code has exceeded its quota of allotted resources, **false** otherwise.

currentCulture: The **language code identifier (LCID)** to be used for the thread.

currentUICulture: The language code identifier (LCID) to be used for the user interface.

2.2.2.12   SPUserCodeExecutionPipelineTerminallyFailedException

This exception indicates that a user code request could not be finished, and that the protocol server is not able to accept more **Execute** (section 3.1.5.1) requests.

```
    namespace Microsoft.SharePoint.UserCode
```

```
    {
        class SPUserCodeExecutionPipelineTerminallyFailedException : SPException
        {
        }
    }
```

2.2.2.13 SPUserCodeExecutionPipelineFailedException

This exception indicates that a single user code request could not be finished, but that the protocol server is able to accept more **Execute** (section 3.1.5.1) requests.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeExecutionPipelineFailedException : SPException
    {
    }
}
```

2.2.2.14 SPUserCodeFeatureCallOutContext

A site solution can contain one or more features, each feature can implement a feature receiver. When a site solution is installed, activated, upgraded, deactivated, or uninstalled on the protocol client, each feature receiver in the site solution is remotely run on the protocol server by calling **Execute** (section 3.1.5.1), with this class specified as the *executionContext* argument. This class contains information that the feature receiver uses to run the request.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeFeatureCallOutContext : SPUserCodeRemoteExecutionContext
    {
        SPFeatureCallOutOperation operation;
        Guid siteId;
        Guid webId;
        SPFeatureScope featureScope;
        Guid featureId;
        string customActionName;
        string[][] parameters;
    }
}
```

operation: An SPFeatureCallOutOperation (section 2.2.1.6) enumeration indicating the type of the current operation.

siteId: The site collection identifier of the current site collection.

webId: The site identifier of the current site (2).

featureScope: An **SPFeatureScope** (section 2.2.1.7) enumeration indicating the scope of this feature callout. This MUST be **SPFeatureScope.Site** or **SPFeatureScope.Web**.

featureId: The feature identifier of the feature that implements the feature receiver being executed.

customActionName: The name of a method to execute when operation is
SPFeatureCallOutOperation.FeatureUpgrading. The protocol server MUST ignore this value if
operation is not **SPFeatureCallOutOperation.FeatureUpgrading**.

parameters: Contains zero or more name/value pairs when operation is
SPFeatureCallOutOperation.FeatureUpgrading. The name/value pairs are parameters to be
passed to **customActionName**. The protocol server MUST ignore this value if operation is not
SPFeatureCallOutOperation.FeatureUpgrading.

2.2.2.15 SPUserCodeRemoteExecutionContext

This is the base class of **SPUserCodeEventHandlerExecutionContext** (section 2.2.2.10) and
SPUserCodeFeatureCallOutContext (section 2.2.2.14) and it contains common information used
by these classes.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeRemoteExecutionContext : SPUserCodeExecutionContext
    {
        string assemblyName;
        string typeName;
    }
}
```

assemblyName: The strong name of the assembly that contains the type specified by the
typeName parameter. This field MUST NOT be **null**. If this field is **null** then the protocol server
MUST throw an **SPUserCodeSolutionExecutionFailedException** (section 2.2.2.16).

typeName: The fully qualified class name of a type, contained in the assembly specified in the
assemblyName field that contains the user code to be executed on the protocol server. This field
MUST NOT be **null**. If this field is **null** then the protocol server MUST throw an
SPUserCodeSolutionExecutionFailedException (section 2.2.2.16).

2.2.2.16 SPUserCodeSolutionExecutionFailedException

This exception indicates that the user code being run on the protocol server has failed.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeSolutionExecutionFailedException : SPException
    {
    }
}
```

2.2.2.17 SPUserCodeValidationFailedException

This exception indicates that a site solution failed validation.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeValidationFailedException : SPException
    {
        string validationErrorMessage;
```

```
            string validationErrorUrl;
        }
    }
```

validationErrorMessage: The error message.

validationErrorUrl: The error URL that contains more information about the failure.

2.2.2.18   SPUserCodeWebPartHttpRequestContext

This class contains information for a Web Part request.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeWebPartHttpRequestContext : SPUserCodeWebPartWrapperContext
    {
        WebPartPageData _webPartPageData;
    }
}
```

_webPartPageData: WebPartPageData (section 2.2.2.34) for the Web Part request.

2.2.2.19   SPUserCodeWebPartHttpResponse

This class contains information for a Web Part response.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeWebPartHttpResponse : SPUserCodeWebPartWrapperContextResponse
    {
        byte[] _output;
        string _htmlContent;
        object PageProperties;
        object PageStateCache;
        string Subtitle;
        string m_assemblyFullName;
        string m_typeFullName;
        UpdatePropertiesWebPartDataSet _updatedPropertiesDataSet;
        WebPartChromeDataSet _partChromeDataSet;
        SPEditorChromeEditorPartSettings _customPropertyGrid;
        SPEditorChromeEditorPartSettings[] _editorPartsSettings;
    }
}
```

_output: This MUST be **null**. This value MUST be ignored by the protocol client.

_htmlContent: HTML (HyperText Markup Language) to be displayed as the content of the Web Part.

PageProperties: Object that the protocol server MAY use as a cache if WebPartPageData.EnablePageProperties was set to **true** by the protocol client. If WebPartPageData.EnablePageProperties was set to **false** the protocol server MUST return **null**.

PageStateCache: Object the protocol server MAY use as a cache.

Subtitle: Value a Web Part can use to render a subtitle.

m_assemblyFullName: The strong name of the assembly that contains the Web Part. This MUST NOT be **null**.

m_typeFullName: The fully qualified class name of the Web Part in the assembly. This MUST NOT be **null**.

_updatedPropertiesDataSet: An **UpdatePropertiesWebPartDataSet** (section 2.2.2.31) to be saved for the Web Part.

_partChromeDataSet: A **WebPartChromeDataSet** (section 2.2.2.32) for Web Part chrome rendering.

_customPropertyGrid: An **SPEditorChromeEditorPartSettings** (section 2.2.2.3) for Web Part customizable and personalizable properties (see [MS-WSSPROG] section 3.1.1.1.1).

_editorPartsSettings: An array of **SPEditorChromeEditorPartSettings** (section 2.2.2.3) for the Web Part.

2.2.2.20   SPUserCodeWebPartImportContext

This class contains information for a Web Part import request.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeWebPartImportContext : SPUserCodeWebPartWrapperContext
    {
        byte _scope;
        string _webPartXml;
    }
}
```

_scope: This MUST be ignored by the protocol server.

_webPartXml: XML that describes the Web Part to be imported as specified in [MS-WPPS] section 2.2.3.3. This value MUST NOT be **null**.

2.2.2.21   SPUserCodeWebPartImportResponse

This class contains information for a Web Part import response.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeWebPartImportResponse
            : SPUserCodeWebPartWrapperContextResponse
    {
        string Subtitle;
        string m_assemblyFullName;
        string m_typeFullName;
        UpdatePropertiesWebPartDataSet _webPartDataSet;
    }
}
```

Subtitle: Value a Web Part can use to render a subtitle.

m_assemblyFullName: The strong name of the assembly that contains the Web Part. This value MUST NOT be **null**.

m_typeFullName: The fully qualified class name of the Web Part in the assembly. This value MUST NOT be **null**.

_webPartDataSet: An **UpdatePropertiesWebPartDataSet** (section 2.2.2.31) to be saved for the Web Part.

2.2.2.22   SPUserCodeWebPartRenderInDesignerContext

This class contains information for a Web Part preview request.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeWebPartRenderInDesignerContext : SPUserCodeWebPartWrapperContext
    {
        WebPartPageData WebPartPageData;
        string ServerRelativeUrl;
    }
}
```

WebPartPageData: WebPartPageData (section 2.2.2.34) for the Web Part request.

ServerRelativeUrl: Server-relative URL of the **Web Part Page** containing the Web Part.

2.2.2.23   SPUserCodeWebPartRenderInDesignerResponse

This class contains information for a Web Part preview response.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeWebPartRenderInDesignerResponse : SPUserCodeWebPartWrapperContextResponse
    {
        string _htmlContent;
    }
}
```

_htmlContent: HTML (HyperText Markup Language) to be displayed as the content of the Web Part.

2.2.2.24   SPUserCodeWebPartWrapperContext

This class is the base class for Web Part requests.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeWebPartWrapperContext : SPUserCodeExecutionContext
    {
        string _pageUrl;
        int _currentCulture;
        int _currentUICulture;
```

```
            WorkerRequestData _workerRequestData;
            SPWebPartManagerData m_webPartManagerData;
            WebPartData WebPartData;
        }
    }
```

_pageUrl: Absolute URL of the Web Part Page containing the Web Part. This value MUST NOT be **null**.

_currentCulture: The language code identifier (LCID) to be used for the thread.

_currentUICulture: The LCID to be used for the user interface.

_workerRequestData: The **WorkerRequestData** (section 2.2.2.36) for the request.

m_webPartManagerData: The **SPWebPartManagerData** (section 2.2.2.30) for the request.

WebPartData: The **WebPartData** (section 2.2.2.33) for the request.

2.2.2.25   SPUserCodeWebPartWrapperContextResponse

This class is the base class for Web Part responses.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeWebPartWrapperContextResponse
    {
    }
}
```

2.2.2.26   SPUserCodeWorkflowActionSandboxExecutionContext

This class contains information for a workflow (2) execution.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeWorkflowActionSandboxExecutionContext : SPUserCodeExecutionContext
    {
        SPUserCodeWorkflowContext Context;
        Hashtable Parameters;
        string AssemblyName;
        string ClassName;
        string FunctionName;
    }
}
```

Context: An **SPUserCodeWorkflowContext** (section 2.2.2.27) denoting the actual workflow (2) context object.

Parameters: A **Hashtable** (section 2.2.2.2) from string to string that contains one key/value pair for each set of parameter names and their values passed to the function specified by the **FunctionName** member.

AssemblyName: The strong name of the assembly that contains the workflow (2). This value MUST NOT be **null**.

ClassName: The fully qualified class name of the workflow (2) in the assembly. This value MUST NOT be **null**.

FunctionName: The function name that will be invoked upon execution.

2.2.2.27   SPUserCodeWorkflowContext

This class contains summary information about a specified **workflow instance** that can be executed remotely.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeWorkflowContext
    {
        string SiteUrl;
        string WebUrl;
        Guid ListId;
        Int32 ItemId;
        Int32 AssociatorUserId;
        Int32 InitiatorUserId;
        string AssociationCategoryName;
        string AssociationTitle;
        DateTime LastRunDateTime;
        DateTime StartedDateTime;
        Guid WorkflowInstanceId;
        string WorkflowStatusUrl;
        string ItemName;
        bool AllowManual;
        bool PermissionsManual;
        bool AutoStartCreate;
        bool AutoStartChange;
        bool AllowDefaultContentApproval;
        string CurrentWebUrl;
        string CurrentItemUrl;
    }
}
```

SiteUrl: The **full URL** of the site collection containing the workflow (2).

WebUrl: The server-relative URL of the site (2) containing the workflow (2).

ListId: The associated list identifier.

ItemId: The list item row identifier.

AssociatorUserId: The **user identifier** of the **workflow association** author.

InitiatorUserId: The user identifier of the user who added or created the workflow (2).

AssociationCategoryName: The category of the workflow association.

AssociationTitle: The title of the workflow association.

LastRunDateTime: The last date and time at which the workflow (2) ran. The type is specified in [MS-NRBF] section 2.1.1.5.

StartedDateTime: The date and time at which the workflow (2) began. The type is specified in [MS-NRBF] section 2.1.1.5.

WorkflowInstanceId: The workflow instance identifier.

WorkflowStatusUrl: The URL of the workflow (2) status page.

ItemName: The name of the **workflow task**.

AllowManual: This is **true** if the workflow association allows manual start, **false** otherwise.

PermissionsManual: This is **true** if the workflow association requires permission for manual start, **false** otherwise.

AutoStartCreate: This is **true** if the workflow MUST start when a list item gets created, **false** otherwise.

AutoStartChange: This is **true** if the workflow MUST start when a list item gets changed, **false** otherwise.

AllowDefaultContentApproval: This is **true** if and only if the base template allows default content approval and the current workflow association identifier is equal to the default **workflow identifier** for content approval of the associated list (1), **false** otherwise. This MUST be **false** for non-list workflows.

CurrentWebUrl: The URL of the workflow's site (2).

CurrentItemUrl: The URL of the workflow's current item. This may be empty.

2.2.2.28   SPUserToken

This class indicates the user on whose behalf the user code is run on the protocol server.

```
namespace Microsoft.SharePoint
{
    class SPUserToken
    {
        byte[] m_token;
    }
}
```

m_token: A variable-length structure associated with a user. The format is specified in [MS-WSSFO] section 2.2.4.2.

2.2.2.29   SPWebEventProperties

This class contains information that an event receiver can use to process a site (2) scoped event (2).

```
namespace Microsoft.SharePoint
{
    class SPWebEventProperties : SPEventPropertiesBase
    {
```

```
                  string m_userDisplayName;
                  string m_userLoginName;
                  string m_fullUrl;
                  string m_serverRelativeUrl;
                  Guid m_parentWebId;
                  Guid m_webId;
                  string m_newServerRelativeUrl;
          }
     }
```

m_userDisplayName: The display name of the user that triggered the event (2).

m_userLoginName: The login name of the user that triggered the event (2).

m_fullUrl: The absolute URL of the site (2) in which the event (2) fired.

m_serverRelativeUrl: The server-relative URL of the site (2) in which the event (2) fired.

m_parentWebId: The site identifier of the parent site (2) if **SPEventPropertiesBase.eventType** field (section 2.2.2.4) is **SPEventReceiverType.WebAdding** (section 2.2.1.5), otherwise this MUST be ignored by the protocol server.

m_webId: The site identifier of the site (2) in which the event (2) was triggered.

m_newServerRelativeUrl: The server-relative URL of the site (2) if **SPEventPropertiesBase.eventType** field (section 2.2.2.4) is **SPEventReceiverType.WebMoved** (section 2.2.1.5), otherwise this MUST be ignored by the protocol server.

2.2.2.30 SPWebPartManagerData

This class contains Web Part manager information for a Web Part.

```
   namespace Microsoft.SharePoint.UserCode
   {
       class SPWebPartManagerData
       {
           string PageUrl;
           string ServerRelativeUrl;
           byte PageView;
           byte InitialUserMode;
           bool ForRender;
           bool CanCustomizePages;
           bool CanPersonalizeWebParts;
           bool CanAddDeleteWebParts;
           bool InDesignMode;
       }
   }
```

PageUrl: The full URL of the Web Part Page containing the Web Part.

ServerRelativeUrl: The server-relative URL of the Web Part Page containing the Web Part.

PageView: Specifies if the Web Part Page is in **shared view** or **personal view**. If the value is set to 0, then the Web Part Page is in shared view; if the value is set to 1, then the Web Part Page is in personal view.

InitialUserMode: Determines if the customizable properties and personalizable properties (see [MS-WSSPROG] section 3.1.1.1.1) can be updated. If **InitialUserMode** is 1 then both the customizable properties and personalizable properties can be updated. If **InitialUserMode** is 2 then only the personalizable properties can be updated.

ForRender: This is **true** if the Web Part is being rendered as part of a browser request, **false** otherwise.

CanCustomizePages: This is **true** if the **current user** has permission to set customizable properties (see [MS-WSSPROG] section 3.1.1.1.1), **false** otherwise.

CanPersonalizeWebParts: This is **true** if the current user has permission to set personalizable properties (see [MS-WSSPROG] section 3.1.1.1.1), **false** otherwise.

CanAddDeleteWebParts: This is **true** if the current user has permission to add or delete Web Parts from the Web Part Page, **false** otherwise.

InDesignMode: This is **true** if the Web Part is in design mode, **false** otherwise.

2.2.2.31   UpdatePropertiesWebPartDataSet

This class contains **Web Part properties** that were updated as part of a Web Part request.

```
namespace Microsoft.SharePoint.UserCode
{
    class UpdatePropertiesWebPartDataSet
    {
        byte[] _allUserProperties;
        byte[] _perUserProperties;
        string _webPartIdProperty;
        bool _safeAgainstScript;
    }
}
```

_allUserProperties: Serialized Web Part properties (section 2.2.3.1) representing 0 or more customizable properties (see [MS-WSSPROG] section 3.1.1.1.1) on the Web Part.

_perUserProperties: Serialized Web Part properties (section 2.2.3.1) representing 0 or more personalizable properties (see [MS-WSSPROG] section 3.1.1.1.1) on the Web Part.

_webPartIdProperty: The identifier of the Web Part.

_safeAgainstScript: This is **true** if the Web Part allows users to add ECMAScript, as specified by [ECMA-262], into its properties, **false** otherwise.

2.2.2.32   WebPartChromeDataSet

This class contains an array of **WebPartVerbData** (section 2.2.2.35) to display on the Web Part chrome.

```
namespace Microsoft.SharePoint.UserCode
{
    class WebPartChromeDataSet
    {
        WebPartVerbData[] WebPartVerbList;
    }
```

```
    }
```

WebPartVerbList: An array of **WebPartVerbData** (section 2.2.2.35) to display on the Web Part chrome.

2.2.2.33   WebPartData

This class contains Web Part data for the Web Part request.

```
namespace Microsoft.SharePoint.UserCode
{
    class WebPartData
    {
        Guid StorageKey;
        string ZoneId;
        int ZoneIndex;
        bool IsClosed;
        bool IsPageStatic;
        byte ChromeState;
        string AssemblyFullName;
        string TypeFullName;
        byte[] AllUsersProperties;
        byte[] PerUserProperties;
        string[][] UnverifiedProperties;
        string WebPartIdProperty;
        byte WebPartType;
        string NamingContainerID;
        string NamingContainerUniqueID;
        bool EnablePagePropertiesForWebPart;
    }
}
```

StorageKey: The **Web Part type identifier** of the Web Part. This value MUST NOT be **null**.

ZoneID: The **Web Part zone identifier** of the Web Part zone that contains the Web Part. This field MUST be ignored by the protocol server if **IsPageStatic** is **true**.

ZoneIndex: The **Web Part zone index** of the Web Part. This field MUST be ignored by the protocol server if **IsPageStatic** is **true**.

IsClosed: This is **true** if the Web Part is closed on the Web Part Page, **false** if the Web Part is open on the Web Part Page.

IsPageStatic: This is **true** if the Web Part is not in a Web Part zone, **false** otherwise.

ChromeState: A **PartChromeState** (section 2.2.1.2) specifies whether the Web Part and its Web Part chrome are in a normal state or in a minimized state.

AssemblyFullName: The strong name of the assembly of the Web Part.

TypeFullName: The fully qualified class name used to identify the Web Part.

AllUsersProperties: Serialized Web Part properties (section 2.2.3.1) that represent 0 or more customizable properties (see [MS-WSSPROG] section 3.1.1.1.1) on the Web Part.

PerUserProperties: Serialized Web Part properties (section 2.2.3.1) that represent 0 or more personalizable properties (see [MS-WSSPROG] section 3.1.1.1.1) on the Web Part.

UnverifiedProperties: Serialized Web Part properties (section 2.2.3.1) that are shared and not personalizable (see [MS-WSSPROG] section 3.1.1.1.1) on the Web Part.

WebPartIdProperty: The HTML (HyperText Markup Language) ID attribute of the Web Part.

WebPartType: Indicates whether customization or personalization is in effect.

Value	Description
1	The **AllUsersProperties** and **PerUserProperties** apply to all users.
2	The **AllUsersProperties** apply to all users and the **PerUserProperties** apply only to the current user.
3	The **AllUsersProperties** and **PerUserProperties** only apply to the current user.

NamingContainerID: Identifier of the container control for the Web Part. This field MUST be ignored by the protocol server if **IsPageStatic** is **true**.

NamingContainerUniqueID: Unique identifier of the container control the Web Part is contained in. The **NamingContainerUniqueID** MAY be used to generate a unique identifier in HTML (HyperText Markup Language) rendering by prepending the **NamingContainerUniqueID** to the identifier field and then replacing all occurrences of the **WebPartPageData.PageIDSeparator** with an underscore character. This field MUST be ignored by the protocol server if **IsPageStatic** is **true**.

EnablePagePropertiesForWebPart: If this field is **true**, the protocol server MAY use or save **PageProperties** in the **SPUserCodeWebPartHttpResponse** that change based on Web Part data. If this field is **false**, the protocol server MUST NOT use or save any **PageProperties** in the **SPUserCodeWebPartHttpResponse** that change based on Web Part data.

2.2.2.34   WebPartPageData

This class contains Web Part Page data for the Web Part request.

```
namespace Microsoft.SharePoint.UserCode
{
    class WebPartPageData
    {
        char PageIDSeparator;
        string SPWebPartManagerID;
        string SPWebPartManagerUniqueID;
        bool EnablePageProperties;
        object PageProperties;
        object PageStateCache;
        string FormID;
        string FormUniqueID;
        string ZoneId;
        bool ZoneAllowPersonalization;
        bool ZoneAllowCustomization;
        string ZoneHeaderText;
        bool ZoneAllowLayoutChange;
        byte ZonePartChromeType;
        byte ZoneLayoutOrientation;
        string ZoneContainerWidth;
```

```
        bool RenderToolPane;
        bool EditorPartApplyCalled;
        bool EditorPartSyncChangesCalled;
        string ToolPaneParentID;
        string ToolPaneParentUniqueID;
        string ToolPaneID;
    }
}
```

PageIDSeparator: The character that MUST be replaced with an underscore character when rendering HTML (HyperText Markup Language) identifier attributes.

SPWebPartManagerID: Identifier of the Web Part manager on the Web Part Page.

SPWebPartManagerUniqueID: Used by the protocol server to generate a unique identifier in HTML (HyperText Markup Language) rendering if **WebPartData.IsStatic** (section 2.2.2.34) is false. The unique identifier MUST be generated by pre-pending **SPWebPartManagerUniqueID** to the Web Part identifier field and then replacing all occurrences of PageIDSeparator in the Web Part identifier with an underscore character.

EnablePageProperties: This is **true** if the protocol server can use the **PaqeProperties** field, **false** otherwise.

PageProperties: Used to persist information between **POST** requests to the same Web Part. The protocol server MUST NOT use the **PageProperties** if **EnablePageProperties** is **false**. The **PageProperties** is the value returned by the protocol server from the previous request using **SPUserCodeWebPartHttpResponse.PageProperties** (section 2.2.2.19). This object is serialized according to [MS-NRTP] section 2.2.2.1.

PageStateCache: Used to persist information between **POST** requests to the same Web Part. The **PageStateCache** is the value returned by the protocol server from the previous request using **SPUserCodeWebPartHttpResponse.PageStateCache** (section 2.2.2.19). This object is serialized according to [MS-NRTP] section 2.2.2.1.

FormID: Identifier of the HTML form tag that contains the Web Part.

FormUniqueID: Unique identifier of the HTML form element that contains the Web Part. The **FormUniqueID** can be used to generate an identifier to the Form element for use in HTML by replacing all occurrences of the **PageIDSeparator** character with an underscore.

ZoneId: The Web Part zone identifier of the Web Part zone that contains the Web Part. The protocol server MUST ignore this field if **WebPartData.IsStatic** (section 2.2.2.34) is **true**.

ZoneAllowPersonalization: This is **true** if Web Parts within the Web Part zone are allowed to update personalizable properties (see [MS-WSSPROG] section 3.1.1.1.1), **false** otherwise. The protocol server MUST ignore this field if **WebPartData.IsStatic** (section 2.2.2.34) is **true**.

ZoneAllowCustomization: This is **true** if Web Parts within the Web Part zone are allowed to update customizable properties (see [MS-WSSPROG] section 3.1.1.1.1), **false** otherwise. The protocol server MUST ignore this field if **WebPartData.IsStatic** (section 2.2.2.34) is **true**.

ZoneHeaderText: Text that the Web Part zone can display when the Web Part Page is in a display mode where the Web Part zones are visible. The protocol server MUST ignore this field if **WebPartData.IsStatic** (section 2.2.2.34) is **true**.

ZoneAllowLayoutChange: This is **true** if the Web Part zone allows the layout of Web Parts it contains to change, **false** otherwise. Layout can change by moving a Web Part into or out of the Web Part zone, rearranging the relative ordering of Web Parts within the Web Part zone, or closing, deleting, minimizing or restoring a Web Part within the Web Part zone. The protocol server MUST ignore this field if **WebPartData.IsStatic** (section 2.2.2.34) is **true**.

ZonePartChromeType: A **PartChromeType** (section 2.2.1.3) that specifies the type of Web Part chrome to render around the Web Part in the Web Part zone. The protocol server MUST ignore this field if **WebPartData.IsStatic** (section 2.2.2.34) is **true**.

ZoneLayoutOrientation: Orientation of Web Parts within the Web Part zone. The protocol server MUST ignore this field if WebPartData.IsStatic (section 2.2.2.34) is **true**.

Value	Description
0	Horizontal – Web Parts will be arranged horizontally.
1	Vertical – Web Parts will be arranged from top to bottom.

ZoneContainerWidth: The expected width of the Web Part zone. This field can take on any valid **CSS** width value. The protocol server MUST ignore this field if **WebPartData.IsStatic** (section 2.2.2.34) is **true**.

RenderToolPane: Indicates if **_customPropertyGrid** and **_editorPartsSettings** fields are used in the **SPUserCodeWebPartHttpResponse**. The protocol server MUST ignore this field if **WebPartData.IsStatic** (section 2.2.2.34) is **true**.

EditorPartApplyCalled: When **true**, properties in the **_customPropertyGrid** and **_editorPartSettings** MAY be set on the Web Part.

EditorPartSyncChangesCalled: When **true**, any cached values for **_customPropertyGrid** and **_editorPartSettings** SHOULD be discarded.

ToolPaneParentUniqueID: Used to generate a unique identifier in HTML (HyperText Markup Language) rendering for the Web Part toolpane by prepending the **ToolPaneParentUniqueID** to the **ToolPaneParentID** field and then replacing all occurrences of **PageIDSeparator** with an underscore character. The Web Part toolpane is the parent control for the **_customPropertyGrid**.

ToolPaneParentID: HTML identifier of the Web Part toolpane parent control.

ToolPaneID: HTML identifier of the Web Part toolpane.

2.2.2.35   WebPartVerbData

This class contains Web Part verb data for display in the Web Part chrome. Web Part verbs provide information about what actions can be performed on the Web Part.

```
namespace Microsoft.SharePoint.UserCode
{
    class WebPartVerbData
    {
        string ID;
        string Text;
        string Description;
        string ClientClickHandler;
        bool Enabled;
```

```
        bool Checked;
        string ImageUrl;
    }
}
```

ID: Identifier of the Web Part verb.

Text: Text of the Web Part verb.

Description: Description of the Web Part verb.

ClientClickHandler: Contains the function name for the ECMAScript function, as specified by [ECMA-262] (section 4.3.24), to be invoked when the Web Part verb is executed.

Enabled: Enabled state of the Web Part verb. The protocol client can use this value to determine if the Web Part verb should be displayed as enabled.

Checked: Checked stated of the Web Part verb. The protocol client can use this value to determine if a check mark should be displayed with the Web Part verb.

ImageUrl: URL to an image that can be used to represent the Web Part verb.

2.2.2.36 WorkerRequestData

This class contains HTTP data for the Web Part request.

```
namespace Microsoft.SharePoint.UserCode
{
    class WorkerRequestData
    {
        string[] _serverVariableKeys;
        string[] _serverVariableValues;
        string[][] _unknownHeaders;
        string[] _knownHeaders;
        byte[] _inputStream;
        string _filePath;
        string _queryString;
        string _rawUrl;
        string _pathInfo;
        bool _haveSPContext;
        Guid _spContextListId;
        int _spContextItemId;
        bool _spContextIsDesignTime;
    }
}
```

_serverVariableKeys: An array of keys that correspond to the elements in the**_serverVariableValues** array to create key/value pairs that provide information about the protocol client's front-end Web server and its current page request.

_serverVariableValues: An array of values that correspond with the elements in the **_serverVariableKeys** array to create key/value pairs that provide information about the protocol client's front-end Web server and its current page request.

_unknownHeaders: A two dimensional array of non-standard HTTP Header values. The first dimension contains the name of the header; the second dimension contains the value of the header.

_knownHeaders: An array of standard HTTP Headers, the header stored at each index of the array is specified in the following table:

Index	HTTP Header
0	Cache-Control
1	Connection
2	Date
3	Keep-Alive
4	Pragma
5	Trailer
6	Transfer-Encoding
7	Upgrade
8	Via
9	Warning
10	Allow
11	Content-Length
12	Content-Type
13	Content-Encoding
14	Content-Language
15	Content-Location
16	Content-MD5
17	Content-Range
18	Expires
19	Last-Modified
20	Accept
21	Accept-Charset
22	Accept-Encoding
23	Accept-Language
24	Authorization
25	Cookie

Index	HTTP Header
26	Expect
27	From
28	Host
29	If-Match
30	If-Modified-Since
31	If-None-Match
32	If-Range
33	If-Unmodified-Since
34	Max-Forwards
35	Proxy-Authorization
36	Referer
37	Range
38	TE
39	User-Agent

_inputStream: HTTP body for the Web Part request.

_filePath: Server-relative URL of the Web Part Page containing the Web Part.

_queryString: This MUST be ignored by the protocol server.

_rawUrl: Full URL of the Web Part Page containing the Web Part.

_pathInfo: The portion of the **_rawUrl** field that is not included as part of the **_filePath**.

_haveSPContext: This is **true** if the _ **spContextListId**, **_spContextItemId**, and **_spContextIsDesignTime** have valid data. If the value is **false** the _ **spContextListId**, **_spContextItemId**, and **_spContextIsDesignTime** fields MUST be ignored by the protocol server.

_spContextListId: The current list identifier for the Web Part request.

_spContextItemId: The current list item identifier for the Web part request.

_spContextIsDesignTime: This is **true** if the current request is rendering for design time, **false** otherwise.

2.2.3 Complex Types

This section specifies complex types passed between the protocol client and protocol server.

2.2.3.1 Serialized Web Part Properties

The format of the records, structures, and tokens used by the Web Part properties are specified in this section.

2.2.3.1.1 Format of the Web Part Properties Record

The **Web Part Properties Record** is a serialized representation of Web Part properties, as follows:

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Header (5 bytes)																														
Header (cont.)								Checksum (variable)																							
Major version (2 bytes)																Minor version (2 bytes)															
Number of Segments (2 bytes)																Segments (variable)															

Header (5 bytes): A 40-bit header. This MUST be 0xFF01142B00.

Checksum (variable): 7BitEncodedInt structure (section 2.2.3.1.8) for the checksum. The checksum is calculated as 3 + (2 * Number of **Segments**) + Number of **Segment Data Records**.

Major version (2 bytes): A 16-bit unsigned integer. This MUST be 0x0202.

Minor version (2 bytes): A 16-bit unsigned integer. This MUST be 0x0203.

Number of Segments (2 bytes): A 16-bit unsigned integer. The high byte MUST be 0x02. The low byte is the number of Segment Records (section 2.2.3.1.2).

Segments (variable): Segment Records (section 2.2.3.1.2). The Number of Segment Records (section 2.2.3.1.2) MUST match the Number of Segments indicated by the low byte of the Number of Segments field.

2.2.3.1.2 Format of the Segment Record

The **Segment Record** is used to store a segment's **Segment Data Records**, as follows:

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
SegmentType (3 bytes)																								Number of Segment Data Records (variable)							
Segment Data Records (variable)																														

SegmentType (3 bytes): A 24-bit value that identifies the type of segment.

Release: Sunday, December 19, 2010

Segment Type	Description
0x010000	**Personalizable Properties**. This segment contains zero or more **Property Name Record**, **Property Name Value** pairs.
0x010100	**Non-Personalizable Properties**. This segment contains zero or more **Property Name Record**, **Property Name Value** pairs.
0x010200	**Private Properties**. This segment contains zero or more **Property Name Record**, **Property Name Value** pairs.
0x010300	**Attached Properties**. This segment contains zero or more **Property Name Record**, **Property Name Value** pairs.
0x010400	**Link Map**. This segment contains zero or more **Property Value Index Records**.

Number of Segment Data Records (variable): Indicates the number of **Segment Data Records**. If the first byte is 0x66 there are no **Segment Data Records** and the following byte is the beginning of the next **Segment Record** (section 2.2.3.1.2). If the first byte is 0x02 the next byte is the number of **Segment Data Records** in the segment.

Segment Data Records (variable): Data for the segment. The **SegmentType** is used to determine the format of the segment data.

2.2.3.1.3   Format of Property Name Record

Property Name Records are used to store the name of a property.

The records are stored as **Property Name Structures** (section 2.2.3.1.6).

2.2.3.1.4   Format of Property Value Record

Property Value Records are used to store the value of a property.

The records are stored as **Property Value Structures** (section 2.2.3.1.7).

2.2.3.1.5   Format of the Property Value Index Record

Property Value Index used to reference a **Property Value Record**.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Property Value Index (variable)																															
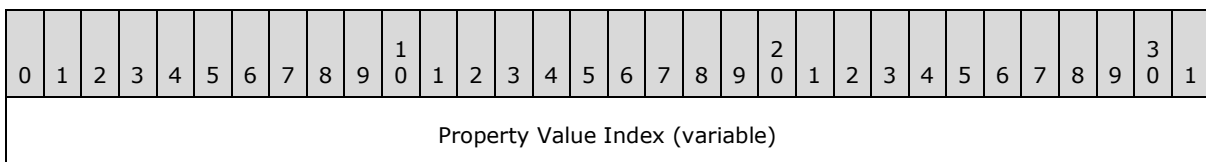
Property Value Index (variable): Index of the **Property Value Record**. The **Property Value index** is stored as a **7BitEncodedInt** (section 2.2.3.1.8).

The location of a **Property Value Record Data** can be found using the **Property Value Index** as follows:

Begin with an initial count value of 3, for example `"int count = 3;"`.

From the beginning of the Serialized Web Part Properties seek forward adding 2 to the count for each **Segment Record** and 1 to the count for each **Segment Data Record** read.

The **Property Value Record** is at the seek location where the count is equal to the **Property Value Index**

2.2.3.1.6   Format of Property Name Structure

Property Name Structures are used to store the name of a property.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Property Name Type (1 byte)								Property Name Data (variable)																							

Property Name Type (1 byte): Identifies the format of the **Property Name Data**.

Property Name Data (variable): The **Property Name Data** is in the format specified by the Property Name Type Field.

Property Name Type	Property Name Data
0x02	**TokenizedStringId** (variable). Data is a **7BitEncodedInt** structure (section 2.2.3.1.8).
0x05	**StringName** (variable). Data is a **String structure**.

2.2.3.1.7   Format of Property Value Structure

Property Value Structures are used to store the value of a property.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Property Value Type (1 byte)								Property Value Data (variable)																							

Property Value Type (1 byte): Indicates **Property Type** and how to decode the **Property Value Data**.

Property Value Data (variable): The **Property Value Data** is in the format specified by the **Property Value Type** field.

Property Value Type	Property Value Data
0x01	**Int16** (2 bytes). Data is written least significant bit first.
0x02	**Int32** (variable). Data is in 7BitEncodedInt structure (section 2.2.3.1.8).
0x03	**Byte** (1 byte)
0x04	**Char** (variable). UTF-8 encoded character.
0x05	**String** (variable). Data is in String structure (section 2.2.3.1.13).

Property Value Type	Property Value Data
0x07	**Double** (8 bytes). Double-precision 64 bit [IEEE754] format.
0x09	**Color** (variable). An **Int32** Color value stored as a **7BitEncodedInt** structure (section 2.2.3.1.8) The components of the **Int32** Color value are: Alpha = 0xFF & (value >> 0x18); Red = 0xFF & (value >> 0x10); Green = 0xFF & (value >> 0x08); Blue = 0xFF & (value);
0x0A	**Tokenized Color** (variable). Tokenized Color stored as a **7BitEncodedInt** structure (section 2.2.3.1.8).
0x0B	**IntEnum** (variable). Data is in IntEnum structure (section 2.2.3.1.9).
0x0C	**EmptyColor** (0 bytes)
0x0F	**Pair** (variable). A Pair of alternating **Property Name** (section 2.2.3.1.6) and **Property Value** structures (section 2.2.3.1.7).
0x1B	**Unit** (variable). Data is in Unit structure (section 2.2.3.1.10).
0x1C	**EmptyUnit** (0 bytes)
0x28	**StringFormatted** (variable). Data is in **StringFormatted** structure (section 2.2.3.1.11).
0x64	**Null** (0 bytes)
0x65	**EmptyString** (0 bytes)
0x66	**ZeroInt32** (0 bytes)
0x67	**Boolean true** (0 bytes)
0x68	**Boolean false** (0 bytes)

2.2.3.1.8 7BitEncodedInt Structure

To save space when persisting integer values the **7bitEncodedInt** structure is used.

The value is written out 7 bits at a time starting with the least significant bits. If the value will not fit in 7 bits the high bit of the byte is set to indicate there is another byte of data to be written. The value is then shifted 7 bits to the right and the next byte is written. If the value will fit in the seven bits the high byte is not set and it signals the end of the structure.

The integer 0x7D would be stored as follows:

The binary representation is 1111101. Because this number can be represented in 7 digits the high bit is not set

0	1	2	3	4	5	6	7
0	1	1	1	1	1	0	1

For a larger value such as 0x123 it is written as follows:

The binary representation is 100100011. The first 7 LSB bits are 0100011 so this is written with the high bit of the first byte set. The remaining bits which are 10 in binary can fit within the next seven bits so the high bit is not set on the second byte.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5
1	0	1	0	0	0	1	1	0	0	0	0	0	0	1	0

2.2.3.1.9   IntEnum Structure

Structure used for writing out an enumeration.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Serialized Type (variable)																															
Value (variable)																															

Serialized Type (variable): The Type information for the enum stored as a **Serialized Type Structure** (section 2.2.3.1.12).

Value (variable): 7BitEncodedInt (section 2.2.3.1.8) representing the value for the enum.

2.2.3.1.10   Unit Structure

Structure used to represent a length of measurement.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Value (8 bytes)																															
Value (continued)																															
UnitType (variable)																															

Value (8 bytes): Value of the unit in double-precision 64 bit [IEEE754] format.

UnitType (variable): Type of unit stored as a **7BitEncodedInt** (section 2.2.3.1.8). The **UnitType** MUST be one of the following:

Unit Type	Unit
0x01	Pixel
0x02	Point
0x03	Pica

Unit Type	Unit
0x04	Inch
0x05	Mm
0x06	Cm
0x07	Percentage
0x08	Em
0x9	Ex

2.2.3.1.11 StringFormatted Structure

Structure used for writing StringFormatted property types.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Serialized Type (variable)																															
Value (variable)																															

Serialized Type (variable): Type information for the property in **Serialized Type Structure** (section 2.2.3.1.12).

Value (variable): String structure (section 2.2.3.1.13) containing a string representation for the value of the property.

2.2.3.1.12 Serialized Type Structure

Structure used for type information.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
TypeRef Type (1 byte)								Type Data (variable)																							

TypeRef Type (1 byte): Indicates how to decode the **Type Data**.

Type Data (variable): Data that represents the type. Data is in the format specified by the TypeRef Type Field.

TypeRef Type	Type Data
0x2B	**WellKnown Type** (variable). **7BitEncodedInt** structure (section 2.2.3.1.8) that MUST be one of the following values 0x00 - **object** 0x01 - **int** 0x02 - **string**

TypeRef Type	Type Data
	0x03 - **bool**
0x2A	**SystemWeb Type** (variable). **String structure** (section 2.2.3.1.13) that contains the fully qualified class name for a class within an assembly with an **assembly name** of **System.Web** and a **public key** of b03f5f7f11d50a3a
0x29	**UnknownType** (variable). **String structure** (section 2.2.3.1.13) that contains the assembly's fully qualified class name.

2.2.3.1.13   String Structure

Structure used for a string.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Length (variable)												Value (variable)																			

Length (variable): 7BitEncodedInt (section 2.2.3.1.8). Length of the Value field.

Value (variable): UTF-8 encoded characters for the string.

2.2.3.1.14   TokenizedStringId

To save space common strings can be saved as a **TokenizedStringId** value.

Token	String
0x00000000	NamespaceWebPart
0x00000001	Dir
0x00000002	Description
0x00000003	Encoding
0x00000004	Title
0x00000005	WebPart
0x00000006	IsIncluded
0x00000007	Zone
0x00000008	ZoneID
0x00000009	PartOrder
0x0000000A	NumberLimit
0x0000000B	FrameState
0x0000000C	Height

Token	String
0x0000000D	Width
0x0000000E	Toolbar
0x0000000F	ContentLink
0x00000010	DisplayName
0x00000011	DataFields
0x00000012	DataQuery
0x00000013	XSLLink
0x00000014	XSL
0x00000015	AllowRemove
0x00000016	AllowMinimize
0x00000017	IsVisible
0x00000018	Namespace
0x00000019	ViewFlag
0x0000001A	DetailLink
0x0000001B	HelpLink
0x0000001C	PartStorage
0x0000001F	PartImageSmall
0x00000020	PartImageLarge
0x00000021	Assembly
0x00000022	TypeName
0x00000025	FrameType
0x00000026	Connections
0x00000027	MissingAssembly
0x00000028	Name
0x00000029	Empty
0x0000002A	Xmlns
0x0000002B	AllowZoneChange
0x0000002C	ParamBindings
0x0000002D	FireInitialRow
0x0000002F	ImageLink

Token	String
0x00000031	CaptureMethod
0x00000032	PostData
0x00000033	Tags
0x00000034	TagIndexes
0x00000035	RenderTags
0x00000036	RenderTagIndexes
0x00000037	LastUpdated
0x00000038	RefreshInterval
0x00000039	LastCached
0x0000003A	Unused11
0x0000003B	Content
0x0000003C	ConnectionID
0x0000003D	XmlSchemaT
0x0000003E	XmlSchemaInstance
0x0000003F	Normal
0x00000040	Minimized
0x00000041	Default
0x00000042	LTR
0x00000043	RTL
0x00000044	None
0x00000045	Standard
0x00000046	TitleBarOnly
0x00000049	Xsi
0x0000004A	Xsd
0x0000004B	NoDefaultStyle
0x0000004C	VerticalAlignment
0x0000004D	HorizontalAlignment
0x0000004E	BackgroundColor
0x0000004F	IsIncludedFilter
0x00000050	XML

Token	String
0x00000051	XMLLink
0x00000052	HeaderCaption
0x00000053	HeaderTitle
0x00000054	HeaderDescription
0x00000055	Image
0x00000056	ContentHasToken
0x00000057	ExportControlledProperties
0x00000058	SourceType
0x00000059	Fields
0x0000005A	NamespaceContentEditorWebPart
0x0000005B	NamespacePageViewerWebPart
0x0000005C	NamespaceImageWebPart
0x0000005D	NamespaceXmlWebPart
0x0000005E	NamespaceDataViewWebPart
0x0000005F	NamespaceListFormWebPart
0x00000060	NamespaceListViewWebPart
0x00000061	NamespacePivotViewWebPart
0x00000062	NamespaceTitleBarWebPart
0x00000063	NamespaceSimpleFormWebPart
0x00000064	NamespaceMembersWebPart
0x00000065	CacheDataStorage
0x00000066	CacheDataTimeout
0x00000067	CacheXslStorage
0x00000068	AlternativeText
0x00000069	DataSourceBindings
0x0000006A	Template
0x0000006B	NamespaceWebPartV3
0x0000006C	ID
0x0000006D	AttachedPropertiesShared
0x0000006E	AttachedPropertiesUser

Token	String
0x0000006F	AllowConnect
0x00000070	AllowEdit
0x00000071	AllowHide
0x00000072	HelpMode
0x00000073	NamespaceUserTasksWebPart
0x00000074	NamespaceUserDocsWebPart
0x00000075	NamespaceAggregationWebPart
0x00000076	QuerySiteCollection
0x00000077	MaxItemsShown
0x00000078	QueryLastModifiedBy
0x00000079	QueryCreatedBy
0x0000007A	QueryCheckedOutBy
0x0000007B	DisplayFolderColumn
0x0000007C	DisplayItemLinkColumn
0x0000007D	TitleUrl
0x0000007E	DisplayType
0x0000007F	MembershipGroupId
0x00000080	AllowClose
0x00000081	AuthorizationFilter
0x00000082	CatalogIconImageUrl
0x00000083	ChromeState
0x00000084	ChromeType
0x00000085	Direction
0x00000086	ExportMode
0x00000087	HelpUrl
0x00000088	Hidden
0x00000089	ImportErrorMessage
0x0000008A	IsClosed
0x0000008B	TitleIconImageUrl
0x0000008C	ZoneIndex

Token	String
0x0000008D	PersonalizableProperties
0x0000008E	NonPersonalizableProperties
0x0000008F	IPersonalizableProperties
0x00000090	AttachedProperties
0x00000091	LinkMap
0x00000093	ViewContentTypeId
0x00000094	CssStyleSheet
0x00000095	ListName

2.2.3.1.15   Tokenized Color

To save space common colors can be saved as a Tokenized Color value.

Token	Color Name
0x01	ActiveBorder
0x02	ActiveCaption
0x03	ActiveCaptionText
0x1C	AliceBlue
0x1D	AntiqueWhite
0x04	AppWorkspace
0x1E	Aqua
0x1F	Aquamarine
0x20	Azure
0x21	Beige
0x22	Bisque
0x23	Black
0x24	BlanchedAlmond
0x25	Blue
0x26	BlueViolet
0x27	Brown
0x28	BurlyWood
0xA8	ButtonFace

Token	Color Name
0xA9	ButtonHighlight
0xAA	ButtonShadow
0x29	CadetBlue
0x2A	Chartreuse
0x2B	Chocolate
0x05	Control
0x06	ControlDark
0x07	ControlDarkDark
0x08	ControlLight
0x09	ControlLightLight
0x0A	ControlText
0x2C	Coral
0x2D	CornflowerBlue
0x2E	Cornsilk
0x2F	Crimson
0x30	Cyan
0x31	DarkBlue
0x32	DarkCyan
0x33	DarkGoldenrod
0x34	DarkGray
0x35	DarkGreen
0x36	DarkKhaki
0x37	DarkMagenta
0x38	DarkOliveGreen
0x39	DarkOrange
0x3A	DarkOrchid
0x3B	DarkRed
0x3C	DarkSalmon
0x3D	DarkSeaGreen
0x3E	DarkSlateBlue

Token	Color Name
0x3F	DarkSlateGray
0x40	DarkTurquoise
0x41	DarkViolet
0x42	DeepPink
0x43	DeepSkyBlue
0x0B	Desktop
0x44	DimGray
0x45	DodgerBlue
0X46	Firebrick
0x47	FloralWhite
0x48	ForestGreen
0x49	Fuchsia
0x4A	Gainsboro
0x4B	GhostWhite
0x4C	Gold
0x4D	Goldenrod
0xAB	GradientActiveCaption
0xAC	GradientInactiveCaption
0x4E	Gray
0X4D	GrayText
0x4F	Green
0x50	GreenYellow
0x0D	Highlight
0x0C	HighlightText
0x51	Honeydew
0x52	HotPink
0x0F	HotTrack
0x10	InactiveBorder
0x11	InactiveCaption
0x12	InactiveCaptionText

Token	Color Name
0x53	IndianRed
0x54	Indigo
0x13	Info
0x14	InfoText
0x55	Ivory
0x56	Khaki
0x57	Lavender
0x58	LavenderBlush
0x59	LawnGreen
0x5A	LemonChiffon
0x5B	LightBlue
0x5C	LightCoral
0x5D	LightCyan
0x5E	LightGoldenrodYellow
0x5F	LightGray
0x60	LightGreen
0x61	LightPink
0x62	LightSalmon
0x63	LightSeaGreen
0x64	LightSkyBlue
0x65	LightSlateGray
0x66	LightSteelBlue
0x67	LightYellow
0x68	Lime
0x69	LimeGreen
0x6A	Linen
0x6B	Magenta
0x6C	Maroon
0x6D	MediumAquamarine
0x6E	MediumBlue

Token	Color Name
0x6F	MediumOrchid
0x70	MediumPurple
0x71	MediumSeaGreen
0x72	MediumSlateBlue
0x73	MediumSpringGreen
0x74	MediumTurquoise
0x75	MediumVioletRed
0x15	Menu
0xAD	MenuBar
0xAE	MenuHighlight
0x16	MenuText
0x76	MidnightBlue
0x77	MintCream
0x78	MistyRose
0x79	Moccasin
0x7A	NavajoWhite
0x7B	Navy
0x7C	OldLace
0x7D	Olive
0x7E	OliveDrab
0x7F	Orange
0x80	OrangeRed
0x81	Orchid
0x82	PaleGoldenrod
0x83	PaleGreen
0x84	PaleTurquoise
0x85	PaleVioletRed
0x86	PapayaWhip
0x87	PeachPuff
0x88	Peru

Token	Color Name
0x89	Pink
0x8A	Plum
0x8B	PowderBlue
0x8C	Purple
0x8D	Red
0x8E	RosyBrown
0x8F	RoyalBlue
0x90	SaddleBrown
0x91	Salmon
0x92	SandyBrown
0x17	ScrollBar
0x93	SeaGreen
0x94	SeaShell
0x95	Sienna
0x96	Silver
0x97	SkyBlue
0x98	SlateBlue
0x99	SlateGray
0x9A	Snow
0x9B	SpringGreen
0x9C	SteelBlue
0x9D	Tan
0x9E	Teal
0x9F	Thistle
0xA0	Tomato
0x1B	Transparent
0xA1	Turquoise
0xA2	Violet
0xA3	Wheat
0xA4	White

Token	Color Name
0xA5	WhiteSmoke
0x18	Window
0x19	WindowFrame
0x1A	WindowText
0xA6	Yellow
0xA7	YellowGreen

3   Protocol Details

3.1   User Code Execution Server Details

The protocol server, on which the user code runs, implements the methods defined in this section to enable protocol clients to run user code remotely.

3.1.1   Abstract Data Model

None.

3.1.2   Timers

None.

3.1.3   Initialization

The interface uses initialization as specified in [MS-NRTP] section 3.2.3.

3.1.4   Higher-Layer Triggered Events

None.

3.1.5   Message Processing Events and Sequencing Rules

The **Microsoft.SharePoint.Administration.ISPUserCodeExecutionHostProxy** interface is comprised of the methods shown in the following table.

Method	Description
Execute	Runs a user code request on the protocol server and returns the result.
Ping	Verifies that the protocol server is available to receive **Execute** messages.

3.1.5.1   Execute

The **Execute** method is called to run a **user code** request on the protocol server.

```
object Execute(Type userCodeWrapperType,
               Guid siteCollectionId,
               SPUserToken userToken,
               string affinityBucketName,
               SPUserCodeExecutionContext executionContext);
```

userCodeWrapperType: The type of object the protocol server uses to execute this user code request. This parameter is a vendor-extensible field and MAY contain the type of any object. <1>

siteCollectionId: A site collection identifier of the current site collection.

userToken: A token (section 2.2.2.28) that identifies the authentication process applied to the current user.

affinityBucketName: A string to suggest a loose affinity between different user code requests. The protocol server can use this string as a suggestion when grouping user code requests together to reuse resources.

executionContext: Context data specific to this user code request. This parameter is a vendor-extensible field and can contain any type that derives from the type **SPUserCodeExecutionContext**.

Return Value: The results of executing this user code request. The type returned is a vendor-extensible field and can be any type that derives from the type **System.Object**.

Exceptions Thrown: If an error occurs while processing this method, one of these exceptions will be thrown:

- **SPUserCodeExecutionPipelineTerminallyFailedException** (section 2.2.2.12)

  The protocol server MUST throw this exception when the protocol server has encountered an unrecoverable and terminal failure. The protocol client MUST NOT send any more Execute requests to the protocol server after this exception has been thrown until the **Ping** method returns **true**.

- **SPUserCodeExecutionPipelineFailedException** (section 2.2.2.13)

  The protocol server MUST throw this exception when the protocol server itself has encountered an error while attempting to execute the user code request.

- **SPUserCodeSolutionExecutionFailedException** (section 2.2.2.16)

  The protocol server MUST throw this exception when the user code being executed on the protocol server has failed.

- **SPUserCodeValidationFailedException** (section 2.2.2.17)

  The protocol server MUST throw this exception when a failure is encountered while validating the assembly group that contains the user code.

  Exceptions thrown by the .NET Remoting Core Protocol as specified in [MS-NRTP] section 2.2.2.

3.1.5.2 Ping

The **Ping** method is called to verify that the protocol server is available to receive **Execute** messages. The protocol server MUST return **true** if it can receive **Execute** messages, otherwise it MUST return **false**.

```
bool Ping();
```

Exceptions Thrown: No exceptions are thrown beyond those thrown by the .NET Remoting Core Protocol as specified in [MS-NRTP] section 2.2.2.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

4   Protocol Examples

4.1   Calling Ping

This example illustrates the message exchange for the remote **Ping** method.

In this sample it is assumed that the `executionHost` is a valid server-activated object that implements the **ISPUserCodeExecutionHostProxy** interface.

```
bool b = executionHost.Ping();
```

The protocol client invokes the method without passing in parameters.

The protocol server is configured to support TCP and the messages are expected to be encoded in [MS-NRBF] section 2.2.3.1. The protocol server is a server-activated object at "tcp://RemotingMachine:49900".

This method returns **true** in the response message to the protocol client.

A sequence diagram for the preceding message exchange pattern is provided in the following figure.
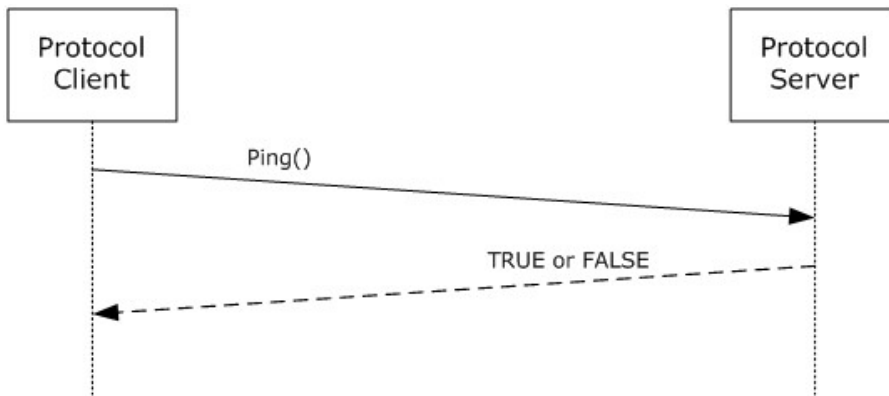
Figure 3: A sequence diagram for the preceding message exchange pattern

The output of the request message is as follows. The interpretation of these message frames are found in [MS-NRTP] section 4.1. There are some differences in this example from the example in [MS-NRTP] section 4.1 and all of those differences are noted as follows.

```
0000   2E 4E 45 54 01 00 00 00 00 00 BB 00 00 00 04 00   .NET......».....
0010   01 01 26 00 00 00 74 63 70 3A 2F 2F 53 48 49 52   ..&...tcp://SHIR
0020   4B 4F 46 32 3A 34 39 39 30 30 2F 53 50 55 43 45   KOF2:49900/SPUCE
0030   78 65 63 75 74 69 6F 6E 48 6F 73 74 06 00 01 01   xecutionHost....
0040   18 00 00 00 61 70 70 6C 69 63 61 74 69 6F 6E 2F   ....application/
0050   6F 63 74 65 74 2D 73 74 72 65 61 6D 00 00         octet-stream..

ContentLength: 187 (0xBB)
Header 1:
    RequestUriHeader
        UriValue: tcp://SHIRKOF2:49900/SPUCExecutionHost
0000   00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00   ................
0010   00 15 11 00 00 00 12 04 50 69 6E 67 12 9B 01 4D   ........Ping.›.M
0020   69 63 72 6F 73 6F 66 74 2E 53 68 61 72 65 50 6F   icrosoft.SharePo
```

```
0030   69 6E 74 2E 41 64 6D 69 6E 69 73 74 72 61 74 69   int.Administrati
0040   6F 6E 2E 49 53 50 55 73 65 72 43 6F 64 65 45 78   on.ISPUserCodeEx
0050   65 63 75 74 69 6F 6E 48 6F 73 74 50 72 6F 78 79   ecutionHostProxy
0060   2C 20 4D 69 63 72 6F 73 6F 66 74 2E 53 68 61 72   , Microsoft.Shar
0070   65 50 6F 69 6E 74 2C 20 56 65 72 73 69 6F 6E 3D   ePoint, Version=
0080   31 34 2E 30 2E 30 2E 30 2C 20 43 75 6C 74 75 72   14.0.0.0, Cultur
0090   65 3D 6E 65 75 74 72 61 6C 2C 20 50 75 62 6C 69   e=neutral, Publi
00A0   63 4B 65 79 54 6F 6B 65 6E 3D 37 31 65 39 62 63   cKeyToken=71e9bc
00B0   65 31 31 31 65 39 34 32 39 63 0B                  e111e9429c.
```

```
SerializationHeaderRecord:
    HeaderId: 0 (0x0)
BinaryMethodCall:
    MessageEnum: 00000014
        NoArgs:                   (...............................1)
        ArgsInline:               (..............................0.)
        ArgsIsArray:              (.............................0..)
        ArgsInArray:              (............................0...)
        NoContext:                (...........................1....)
        ContextInline:            (..........................0.....)
        ContextInArray:           (.........................0......)
        MethodSignatureInArray:   (........................0.......)
        PropertyInArray:          (.......................0........)
        NoReturnValue:            (......................0.........)
        ReturnValueVoid:          (.....................0..........)
        ReturnValueInline:        (....................0...........)
        ReturnValueInArray:       (...................0............)
        ExceptionInArray:         (..................0.............)
        Reserved:                 (000000000000000000..............)
    MethodName:
        Data: Ping
    TypeName:
        Data: Microsoft.SharePoint.Administration.ISPUserCodeExecutionHostProxy,
Microsoft.SharePoint, Version=14.0.0.0, Culture=neutral, PublicKeyToken= 71e9bc e111e9429c
MethodCallArray:
```

The **MethodCallArray** record is empty because the **Ping** method does not take any parameters.

```
0010   2E 4E 45 54 01 00 02 00 00 00 19 00 00 00 00 00   .NET............
ContentLength
    ContentLength: 25 (0x19)
```

```
0000   00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00   ................
0010   00 16 11 08 00 00 01 01 0B                        .........
```

```
ReturnValue:
    PrimitiveTypeEnum: Boolean (0x01)
    Data: TRUE (0x01)
```

5 Security

5.1 Security Considerations for Implementers

This protocol inherits the security considerations described in [MS-NRTP] section 5. Because this protocol specifies no authentication or authorization mechanisms, a protocol server can perform implementation-specific authorization only based on evidence from underlying transport mechanisms. Operational security can be enhanced by restricting transport layer access to the protocol servers to only known protocol clients, such as other machines in the farm.

5.2 Index of Security Parameters

None.

6   Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® SharePoint® Foundation 2010

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 3.1.5.1: In SharePoint Foundation 2010 there is a relationship between the type of object passed to the *userCodeWrapperType* parameter on the **Execute** method and the type of object that is passed to the parameter executionContext, and the type of object that is returned from the Execute message.

The following table shows the relationship between the types passed into the userCodeWrapperType parameter by SharePoint Foundation 2010 to the types that are passed to the *executionContext* parameter, and the type of object that is returned from this method.

Type	Execution Context Object	Return Object
SPUserCodeWebPartWrapper	**SPUserCodeWebPartWrapperContext**	SPUserCodeWebPartWrapperContextResponse
SPUserCodeWorkflowActionWrapper	**SPUserCodeWorkflowActionSandboxExecutionContext**	any Object
SPUserCodeEventHandlerWrapper	**SPUserCodeEventHandlerExecutionContext**	EventResults
SPUserCodeFeatureCallOutWrapper	**SPUserCodeFeatureCallOutContext**	null

In addition to this table the SharePoint Foundation 2010 protocol server can also accept any class that derives from the class **SPUserCodeWrapper**. When an unknown class derived from **SPUserCodeWrapper** is passed to the parameter *userCodeWrapperType*, and then the SharePoint Foundation 2010 protocol server will be able to accept any class that derives from the class **SPUserCodeExecutionContext** for the executionContent and will return any class that derives from system.Object. See [MSDN-SharePointSDK] for information about implementation-specific details of the **SPUserCodeWrapper** class.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index