[MS-SPPTC]:

User Code Execution Protocol

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 Open Specification Promise or the Community Promise. 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. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **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
7/13/2009	0.1	Major	Initial Availability
8/28/2009	0.2	Editorial	Revised and edited the technical content
11/6/2009	0.3	Editorial	Revised and edited the technical content
2/19/2010	1.0	Major	Updated and revised the technical content
3/31/2010	1.01	Editorial	Revised and edited the technical content
4/30/2010	1.02	Editorial	Revised and edited the technical content
6/7/2010	1.03	Editorial	Revised and edited the technical content
6/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
7/23/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
9/27/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	1.04	None	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	1.04	None	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	2.0	Major	Significantly changed the technical content.
4/11/2012	2.0	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	2.0	None	No changes to the meaning, language, or formatting of the technical content.
9/12/2012	2.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	2.1	Minor	Clarified the meaning of the technical content.
2/11/2013	2.2	Minor	Clarified the meaning of the technical content.
7/30/2013	2.3	Minor	Clarified the meaning of the technical content.
11/18/2013	2.3	None	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	2.4	Minor	Clarified the meaning of the technical content.
4/30/2014	2.5	Minor	Clarified the meaning of the technical content.
7/31/2014	2.5	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
10/30/2014	2.5	None	No changes to the meaning, language, or formatting of the technical content.
2/26/2016	3.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The User Code Execution Protocol allows a protocol client to call an execution service on a protocol server to run code.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

absolute URL: The full Internet address of a page or other World Wide Web resource. The absolute URL includes a protocol, such as "http," a network location, and an optional path and file name — for example, http://www.treyresearch.net/.

anonymous user: A user who presents no credentials when identifying himself or herself. The process for determining an anonymous user can differ based on the authentication protocol, and the documentation for the relevant authentication protocol should be consulted.

assembly: A collection of one or more files that is versioned and deployed as a unit. An assembly is the primary building block of a .NET Framework application. All managed types and resources are contained within an assembly and are marked either as accessible only within the assembly or as accessible from code in other assemblies. Assemblies also play a key role in security. The code access security system uses information about an assembly to determine the set of permissions that is granted to code in the assembly.

assembly name: The name of a collection of one or more files that is versioned and deployed as a unit. See also **assembly**.

authenticated user: A built-in security group specified in [MS-WSO] whose members include all users that can be authenticated by a computer.

cascading style sheet (CSS): An extension to **HTML** that enables authors and users of HTML documents to attach style sheets to those documents, as described in [CSS-LEVEL1] and [CSS-LEVEL2]. A style sheet includes typographical information about the appearance of a page, including the font for text on the page.

checked out: A publishing level that indicates that a document has been created and locked for exclusive editing by a user in a version control system.

configuration database: A database that is stored on a back-end database server and contains both persisted objects and site collection metadata for lookup purposes.

content database: A database that is stored on a back-end database server and contains stored procedures, site collections, and the contents of those site collections.

culture name: A part of a language identification tagging system, as described in [RFC1766]. Culture names adhere to the format "<languagecode2>-<country/regioncode2>." If a two-letter language code is not available, a three-letter code that is derived from [ISO-639] is used.

current user: The user who is authenticated during processing operations on a **front-end web server** or a back-end database server.

display name: A text string that is used to identify a principal or other object in the user interface. Also referred to as title.

document library: A type of list that is a container for documents and folders.

event: An action or occurrence to which an application might respond. Examples include state changes, data transfers, key presses, and mouse movements.

event handler: A software routine that executes in response to an event.

event receiver: A structured modular component that enables built-in or user-defined managed code classes to act upon objects, such as list items, **lists (1)**, or content types, when specific triggering actions occur.

farm: A group of computers that work together as a single system to help ensure that applications and resources are available. Also referred to as server farm.

feature: A package of SharePoint elements that can be activated or deactivated for a specific feature scope.

feature identifier: A **GUID** that identifies a **feature**.

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.

feature scope: The scope at which a **feature** can be activated.

field: A container for metadata within a SharePoint list and associated list items.

form digest validation: A type of security validation that helps prevent an attack wherein users are tricked into posting data to a server.

front-end web server: A server that hosts webpages, performs processing tasks, and accepts requests from protocol clients and sends them to the appropriate back-end server for further processing.

full URL: A string of characters in a standardized format that identifies a document or resource on the World Wide Web.

fully qualified class name: A class name that includes namespace information. Use of a fully qualified class name ensures that the class name is treated as unique.

globally unique identifier (GUID): A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the **GUID**. See also universally unique identifier (UUID).

hash: A fixed-size result that is obtained by applying a one-way mathematical function, which is sometimes referred to as a hash algorithm, to an arbitrary amount of data. If the input data changes, the hash also changes. The hash can be used in many operations, including authentication (2) and digital signing.

HTTP method: In an HTTP message, a token that specifies the method to be performed on the resource that is identified by the Request-URI, as described in [RFC2616].

Hypertext Markup Language (HTML): An application of the Standard Generalized Markup Language (SGML) that uses tags to mark elements in a document, as described in [HTML].

Hypertext Transfer Protocol (HTTP): An application-level protocol for distributed, collaborative, hypermedia information systems (text, graphic images, sound, video, and other multimedia files) on the World Wide Web.

language code identifier (LCID): A 32-bit number that identifies the user interface human language dialect or variation that is supported by an application or a client computer.

list: (1) A container within a SharePoint site that stores list items. A list has a customizable schema that is composed of one or more fields.

(2) An organization of a region of cells into a tabular structure in a workbook.

list identifier: A GUID that is used to identify a **list (1)** in a site collection.

list item: An individual entry within a SharePoint list. Each list item has a schema that maps to fields in the list that contains the item, depending on the content type of the item.

list item attachment: A file that is contained within a list item and is stored in a folder in the **list (1)** with the segment "Attachments".

list item identifier: See item identifier.

list server template: A value that identifies the template that is used for a SharePoint list.

login name: A string that is used to identify a user or entity to an operating system, directory service, or distributed system. For example, in Windows-integrated authentication, a login name uses the form "DOMAIN\username".

managed code: Code that is executed by the common language runtime (CLR) environment rather than directly by the operating system. Managed code applications gain CLR services, such as automatic garbage collection, runtime type checking, and security support. These services provide uniform behavior that is independent of platform and language.

personal view: A view of a list that is created by a user for personal use. The view is unavailable to other users.

personal Web Part: A **Web Part** that was added to the personal view of a Web Parts Page and is available only to the user who added it.

public key: One of a pair of keys used in public-key cryptography. The public key is distributed freely and published as part of a digital certificate. For an introduction to this concept, see [CRYPTO] section 1.8 and [IEEE1363] section 3.1.

request identifier: A **GUID** that is used to identify a specific action or procedure that is sent to a protocol server or a protocol client.

server-activated object (SAO): A server object that is created on demand in response to a client request. See also marshaled server object.

server-relative URL: A relative URL that does not specify a scheme or host, and assumes a base URI of the root of the host, as described in [RFC3986].

shared view: A view of a list or Web Parts Page that every user who has the appropriate permissions can see.

site collection: A set of **websites** that are in the same **content database**, have the same owner, and share administration settings. A site collection can be identified by a **GUID** or the **URL** of the top-level site for the site collection. Each site collection contains a top-level site, can contain one or more subsites, and can have a shared navigational structure.

site collection identifier: A GUID that identifies a site collection. In stored procedures, the identifier is typically "@SiteId" or "@WebSiteId". In databases, the identifier is typically "SiteId/tp_SiteId".

site identifier: A GUID that is used to identify a site in a **site collection**.

site solution: A deployable, reusable package that contains a set of features, site definitions, and assemblies that apply to sites, and can be enabled or disabled individually.

SOAP operation: An action that can be performed by a Simple Object Access Protocol (SOAP) service, as described in [SOAP1.1].

strong name: A name that consists of the simple text name, version number, and culture information of an **assembly**, strengthened by a public key and a digital signature that is generated over the assembly.

Uniform Resource Locator (URL): A string of characters in a standardized format that identifies a document or resource on the World Wide Web. The format is as specified in [RFC1738].

user code: Managed code that can be uploaded to a site by a site collection administrator, without approval from the server farm administrator. It cannot access code or data on other site collections.

user identifier: An integer that uniquely identifies a security principal (2) as distinct from all other security principals (2) and site groups within the same site collection.

web application: A container in a configuration database that stores administrative settings and entry-point **URLs** for **site collections**.

Web Part: A reusable component that contains or generates web-based content such as **XML**, **HTML**, and scripting code. It has a standard property schema and displays that content in a cohesive unit on a webpage. See also Web Parts Page.

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.

Web Part Page: An ASP.NET webpage that includes Web Part controls that enable users to customize the page, such as specifying which information to display. Referred to as Web Parts Page in Microsoft SharePoint Foundation 2010.

Web Part property: A configurable characteristic of a Web Part that determines the behavior of the Web Part.

Web Part type identifier: A unique 16-byte value that is assigned to each Web Part type.

Web Part zone: A structured HTML section of a Web Parts Page that contains zero or more Web Parts and can be configured to control the organization and format of those Web Parts.

Web Part zone identifier: A string that identifies a Web Part zone on a Web Parts Page.

Web Part zone index: An integer that specifies the relative position of a Web Part in a Web Part zone. Web Parts are positioned from the smallest to the largest zone index. If two or more Web Parts have the same zone index they are positioned adjacent to each other in an undefined order.

website: A group of related pages and data within a SharePoint site collection. The structure and content of a site is based on a site definition. Also referred to as SharePoint site and site.

workflow: A structured modular component that enables the automated movement of documents or items through a specific sequence of actions or tasks that are related to built-in or user-defined business processes.

workflow association: An association of a workflow template to a specific list or content type.

workflow instance: An instance of a workflow association that performs on a list item the process that is defined in a workflow template.

workflow task: An action or task in a sequence that is related to a built-in or user-defined business process.

XML: The Extensible Markup Language, as described in [XML1.0].

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

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.

[ECMA-262-1999] ECMA International, "Standard ECMA-262 ECMAScript Language Specification", 3rd edition (December 1999), http://www.ecma-international.org/publications/files/ECMA-ST-ARCH/ECMA-262,%203rd%20edition,%20December%201999.pdf

[IEEE754] IEEE, "IEEE 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".

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

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

[MS-WSSFO2] Microsoft Corporation, "Windows SharePoint Services (WSS): File Operations Database Communications Version 2 Protocol".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.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", December 2007, http://msdn.microsoft.com/en-us/library/ms441339.aspx

1.3 Overview

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 of the farm that are responding to Web requests and increases their stability by isolating them from problems caused by the currently 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.

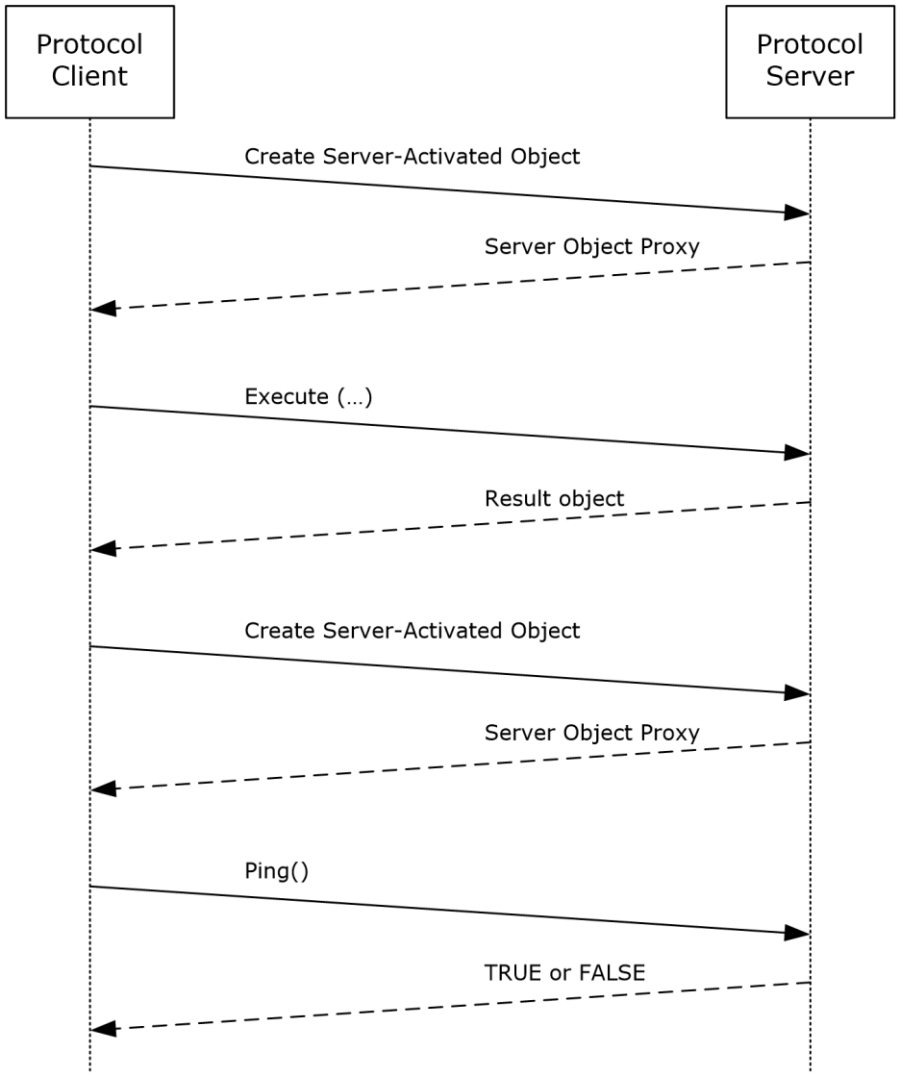


Figure 1 Overview of use of protocol to run a user code request

1.4 Relationship to Other Protocols

This protocol depends on the .NET Remoting Core Protocol described in [MS-NRTP], as shown in the following layering diagram:

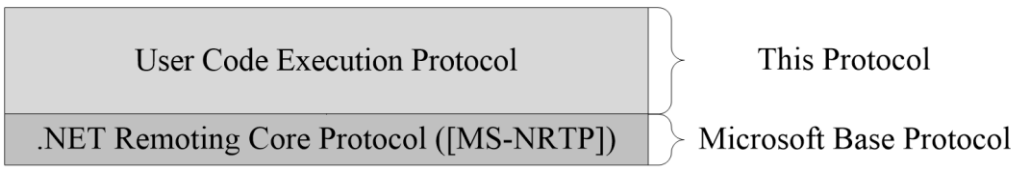


Figure 2 This protocol in relation to other protocols

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 (SAO)**, 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 to be used 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]. This interface is **ISPUserCodeExecutionHostProxy**, as specified in section 3.1.5.

The server-activated object (SAO) 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 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** 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 **website** 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 succeeded and the request will continue.

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

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

CancelWithRedirectUrl: The event 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.

```
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 that is raised before a list item is added to a list (1).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

WebDeleting: Type of event that is raised before a website is deleted.

WebMoving: Type of event that is raised before the URL of a website is changed.

WebAdding: Type of event that is raised before a website is created.

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

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

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

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

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

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

ItemUncheckedOut: Type of event that is raised after a list item is reverted from checked out.

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

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

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

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

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

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

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

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

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

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

WebDeleted: Type of event that is raised after a website is deleted.

WebMoved: Type of event that is raised after the URL of a website is changed.

WebProvisioned: Type of event that is raised before a website is provisioned.

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

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

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

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

ContextEvent: This value is not a type of event that gets raised; instead it is used to identify workflow 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** level.

Site: The feature operates at the site collection level.

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

2.2.1.8 HttpParametersFlags

This enumeration is used to determine the HTTP customized bitmask flags.<1>

```
enum HttpParametersFlags
{
    hpfIgnoreFormHash = 0x01,
    hpfLayoutsPage = 0x02,
    hpfAdminPage = 0x04,
    hpfAuthenticatedRequest = 0x08,
    hpfGlobalAdmin = 0x10,
    hpfDisableListEvents = 0x20,
    hpfSoapRequest = 0x40
}
```

hpfIgnoreFormHash: Indicates that the **HTTP** request should ignore the **form digest validation**.

hpfLayoutsPage: Indicates that the HTTP request is for a Web page that does not require administrative privileges.

hpfAdminPage: Indicates that the HTTP request is for a Web page that requires administration privileges.

hpfAuthenticatedRequest: Indicates that the HTTP request is for an **authenticated user**, as opposed to an **anonymous user**.

hpfGlobalAdmin: Indicates that the HTTP request is an administration operation.

hpfDisableListEvents: Indicates that list (1) events will be disabled for this HTTP request.

hpfSoapRequest: Indicates that the HTTP request is a **SOAP operation**.

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;
            String[][2] changedProperties;
            String errorMessage;
            String redirectUrl;
        }
    }
```

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

changedProperties: SHOULD<2> contain one string value pair for each field of the list item that was changed during the execution of the event handler. It MUST be **NULL** if the **EventScope** (section 2.2.1.1) is not **EventScope.Item** in the **SPUserCodeEventHandlerExecutionContext.eventScope** field (section 2.2.2.12) for which this **EventResults** is returned.

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;
        object[] Keys;
        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 (2) MUST be **null**.

HashCodeProvider: Reserved. The value of this field (2) 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 Guid

This class is used to contain a **GUID**.<3>

```
namespace System
```

```
{
    class Guid
    {
       Int32 _a;
       Int16 _b;
       Int16  c;
       Byte _d;
       Byte _e;
       Byte _f;
       Byte _g;
       Byte  h;
       Byte _i;
       Byte _j;
       Byte_k;
    }
}
```

_a: The first 4 bytes of the GUID.

_b: The next 2 bytes of the GUID.

_c: The next 2 bytes of the GUID.

_d: The next byte of the GUID.

_e: The next byte of the GUID.

_f: The next byte of the GUID.

_g: The next byte of the GUID.

_h: The next byte of the GUID.

_i: The next byte of the GUID.

_j: The next byte of the GUID.

_k: The next byte of the GUID.

2.2.2.4 Nullable<bool>

This class is used to contain a **bool** value type that can also indicate if the **bool** value has been initialized.<4>

```
namespace System
{
    class Nullable<bool>
    {
       bool HasValue;
       bool Value;
    }
}
```

HasValue: true if the **Value** property has been initialized.

Value: The value of the **bool** if the **HasValue** property is **true**.

2.2.2.5 Nullable<Double>

This class is used to contain a **Double** value type that can also indicate if the **Double** value has been initialized.<5>

```
namespace System
{
    class Nullable<Double>
    {
       bool HasValue;
       Double Value;
    }
}
```

HasValue: true if the **Value** property has been initialized.

Value: The value of the **Double** if the **HasValue** property is **true**.

2.2.2.6 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 **Hypertext Markup Language (HTML)** 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 **Body** element, **false** otherwise.

m_allowMinimize: This is **true** if the **m_renderHtml** can be used within Web Part chrome HTML that has the 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.

2.2.2.7 SPEventPropertiesBase

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

```
namespace Microsoft.SharePoint
{
    abstract class SPEventPropertiesBase
    {
        Guid m_siteId;
        bool m marshalOnDemand;
        String m_receiverData;
        SPEventReceiverStatus m_status;
        String m_errorMessage;
        SPEventReceiverType m_eventType;
        String m redirectUrl;
    }
}
```

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

m_marshalOnDemand: MUST be **false**.

m_receiverData: Additional data persisted on behalf of the event receiver implementation to be passed to the event receiver 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_errorMessage: The protocol client MUST set this to **NULL**. If an error message is generated during the event receiver's execution, it 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.

m_redirectUrl: The protocol client MUST set this to **NULL**. If a redirect URL is generated during the event receiver's execution, it will be returned from the protocol server to the protocol client in **EventResults** class (section 2.2.2.1).

2.2.2.8 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**.

2.2.2.9 SPItemEventDataCollection

This class contains information about a list item event.

```
namespace Microsoft.SharePoint
{
    class SPItemEventDataCollection
    {
        bool m_marshalOnDemand;
        bool m_allowChanges;
        String[][2] m_properties;
        String[][2] m_changedProperties;
    }
}
```

m_marshalOnDemand: MUST be **false**.

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

m_properties: Property bag containing one string/value pair for each property associated with the item event.<6>

m_changedProperties: Contains one string/value pair for each field of the list item that was changed during the execution of the event handler.<7>

2.2.2.10 SPItemEventProperties

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

```
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 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 website in which the event occurred.

m_webRelUrl: The **server-relative URL** of the website.

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

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

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

m_beforeProperties: An **SPItemEventDataCollection** (section 2.2.2.9) containing the properties of the list item before the event occurred. The event receiver cannot change the value of these properties so **SPItemEventDataCollection.m_allowChanges** (section 2.2.2.9) MUST be **false**.

m_afterProperties: An **SPItemEventDataCollection** (section 2.2.2.9) containing the properties of the list item after the event occurred. The event handler can change the values of these properties when **SPEventPropertiesBase.m_eventType** (section 2.2.2.7) is **SPEventReceiverType.ItemAdding** (section 2.2.1.5) or **SPEventReceiverType.ItemUpdating** (section 2.2.1.5). **SPItemEventDataCollection.m_allowChanges** (section 2.2.2.9) MUST be **true** in these cases and **false** otherwise.

2.2.2.11 SPListEventProperties

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

```
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;
        Int32 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 website in which the event occurred.

m_webId: The **site identifier** of the website in which the event 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 that is affected by adding, removing, or updating a field (2) in the list (1) if **SPEventPropertiesBase.eventType** field (section 2.2.2.7) is one of the following:

- **FieldAdding**
- **FieldUpdating**
- **FieldDeleting**

- **FieldAdded**
- **FieldUpdated**
- **FieldDeleted** (section 2.2.1.5)

Otherwise this MUST be ignored.

m_fieldName: The internal field name that is affected by adding, removing, or updating a field (2) in the list (1) if **SPEventPropertiesBase.eventType** field is one of the following:

- **FieldAdding**
- **FieldUpdating**
- **FieldDeleting**
- **FieldAdded**
- **FieldUpdated**
- **FieldDeleted**

Otherwise this MUST be ignored.

m_templateId: The **list server template**, as specified in [MS-WSSFO2] 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.

2.2.2.12 SPUserCodeEventHandlerExecutionContext

This class contains information that an event handler uses to execute 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 2.2.2.7). 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 2.2.2.10)
EventScope.List	**SPListEventProperties** (section 2.2.2.11)
EventScope.Web	**SPWebEventProperties** (section 2.2.2.31)

tranLockId: This MUST be ignored by the protocol server.

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

2.2.2.13 SPUserCodeExecutionContext

This class contains the context specific data regarding a specific call to execute 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;
        Nullable<Double> resourceValue;
        Nullable<bool> resourceQuotaExceeded;
        String currentCulture;
        String currentUICulture;
        Guid executingCorrelationId;
        int executingRequestId;
        String HttpRequestMethod;
        String HttpFormDigest;
        uint HttpRequestFlags;
    }
}
```

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

solutionHash: The implementation-specific **hash** of the content of the site solution containing user code.

solutionValidatorsHash: The implementation-specific hash of the site solution validators that validated the site solution containing user code.

solutionValidationErrorUrl: This value represents a URL to redirect to when the last solution validation failed. This value MUST be null when the last solution validation succeeded. This field MAY<8> be present.

solutionValidationErrorMessage: This value is an error message to render when the last solution validation failed and solutionValidationErrorUrl is null. This value MUST be null when the last solution validation succeeded. This field MAY<9> be present.

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: A **Nullable<Double>** (section 2.2.2.5) value indicating the resource usage quota value of this site solution per execution request.

resourceQuotaExceeded: A **Nullable<bool>** (section 2.2.2.4). This is **true** if the site collection containing this site solution with user code has exceeded its quota of allotted resources.

currentCulture: The **culture name** used for the thread executing the user code request.

currentUICulture: The culture name of the user interface used to execute the user code request.

executingCorrelationId: The **request identifier** set by the protocol client for the user code request.

executingRequestId: An identifier established by the protocol server for the user code request.

HttpRequestMethod: The **HTTP method** that was used to initiate the user code request. This field can be **NULL**.

HttpFormDigest: The form digest validation used for the user code request. This field can be **NULL**.

HttpRequestFlags: An **HttpParametersFlags** (section 2.2.1.8) bitmask enumeration.

2.2.2.14 SPUserCodeExecutionPipelineTerminallyFailedException

This exception indicates that a user code request could not be completed, 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
    {
    }
}
```

2.2.2.15 SPUserCodeExecutionPipelineFailedException

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

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

2.2.2.16 SPUserCodeFeatureCallOutContext

A site solution MAY contain one or more features, each feature MAY 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 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 execute the request.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeFeatureCallOutContext : SPUserCodeRemoteExecutionContext
    {
        SPFeatureCallOutOperation operation;
        Guid siteId;
        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.

featureScope: An **SPFeatureScope** (section 2.2.1.7) enumeration indicating the scope of this feature callout. This MUST be "Site" or "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**. Otherwise, the protocol server MUST ignore this value.

parameters: Contains zero or more name-value pairs when the operation is **SPFeatureCallOutOperation.FeatureUpgrading**<10>. The name/value pairs are parameters to be passed to customActionName. Otherwise, the protocol server MUST ignore this value.

2.2.2.17 SPUserCodeRemoteExecutionContext

This is the base class of **SPUserCodeEventHandlerExecutionContext** (section 2.2.2.12) and **SPUserCodeFeatureCallOutContext** (section 2.2.2.16) 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 2.2.2.18).

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 2.2.2.18).

2.2.2.18 SPUserCodeSolutionExecutionFailedException

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

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

2.2.2.19 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.

2.2.2.20 SPUserCodeWebPartHttpRequestContext

This class contains information for a Web Part request.

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

_webPartPageData: WebPartPageData (section 2.2.2.36) for the Web Part request.

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

_htmlContent: HTML 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 2.2.2.33) to be saved for the Web Part.

_partChromeDataSet: A **WebPartChromeDataSet** (section 2.2.2.34) for Web Part chrome rendering.

_customPropertyGrid: An **SPEditorChromeEditorPartSettings** (section 2.2.2.6) for Web Part customizable and personalizable properties (section 3.1.1.1.1).

_editorPartsSettings: An **SPEditorChromeEditorPartSettings** (section 2.2.2.6) for the Web Part.

2.2.2.22 SPUserCodeWebPartImportContext

This class is defined as follows:

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

_scope: MUST be ignored.

_webPartXml: XML that describes the Web Part to be imported, as specified in [MS-WPPS] section 2.2.3.1. This value MUST NOT be **null**.

2.2.2.23 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 2.2.2.33) to be saved for the Web Part.

2.2.2.24 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 2.2.2.36) for the Web Part request.

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

2.2.2.25 SPUserCodeWebPartRenderInDesignerResponse

This class contains information for a Web Part preview response.

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

_htmlContent: HTML to be displayed as the content of the Web Part.

2.2.2.26 SPUserCodeWebPartWrapperContext

This class is the base class for Web Part requests.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeWebPartWrapperContext : SPUserCodeExecutionContext
    {
        String  pageUrl;
        Int32 _currentCulture;
        Int32  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 language code identifier (LCID) to be used for the user interface.

_workerRequestData: The **WorkerRequestData** (section 2.2.2.38) for the request.

m_webPartManagerData: The **SPWebPartManagerData** (section 2.2.2.32) for the request.

WebPartData: The **WebPartData** (section 2.2.2.35) for the request.

2.2.2.27 SPUserCodeWebPartWrapperContextResponse

This class is the base class for Web Part responses.

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

2.2.2.28 SPUserCodeWorkflowActionSandboxExecutionContext

This class adds workflow execution information to the **SPUserCodeExecutionContext**.

```
namespace Microsoft.SharePoint.UserCode
{
    class SPUserCodeWorkflowActionSandboxExecutionContext : SPUserCodeExecutionContext
    {
        SPUserCodeWorkflowContext Context;
        String [][2] Parameters;
        String AssemblyName;
        String ClassName;
        String FunctionName;
    }
}
```

Context: An **SPUserCodeWorkflowContext** (section 2.2.2.29) denoting the actual workflow (2) context object.

Parameters: The collection of parameter names and their values passed to the function specified by the **FunctionName** member.<11>

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.

2.2.2.29 SPUserCodeWorkflowContext

This class contains 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;
    }
}
```

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

WebUrl: The server-relative URL of the website containing the workflow.

ListId: The associated list identifier.

ItemId: The list (2) 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.

AssociationCategoryName: The category of the workflow association.

AssociationTitle: The title of the workflow association.

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

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

WorkflowInstanceId: The workflow instance identifier.

WorkflowStatusUrl: The URL of the workflow status page.

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

2.2.2.30 SPUserToken

This class indicates the user on whose behalf the user code is executed 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-WSSFO2] section 2.2.4.2.

2.2.2.31 SPWebEventProperties

This class contains information that an event handler can use to process a web scoped event.

```
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 website in which the event fired.

m_serverRelativeUrl: The server-relative URL of the website in which the event fired.

m_parentWebId: Identifies the parent website if **SPEventPropertiesBase.eventType** field (section 2.2.2.7) is **SPEventReceiverType.WebAdding** (section 2.2.1.5), otherwise this MUST be ignored.

m_webId: Identifies the website in which the event was triggered.

m_newServerRelativeUrl: The server-relative URL of the website if **SPEventPropertiesBase.eventType** field (section 2.2.2.7) is **SPEventReceiverType.WebMoved** (section 2.2.1.5), otherwise this MUST be ignored.

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

ServerRelativeUrl: 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, the Web Part Page is in shared view; if the value is set to 1, the Web Part Page is in personal view.

InitialUserMode: Determines if the customizable properties and personalizable properties (section 3.1.1.1.1) MAY be updated. If **InitialUserMode** is 1 then both the customizable properties and personalizable properties are allowed to 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.

CanCustomizePages: This is **true** if the **current user** has permission to set customizable properties (section 3.1.1.1.1).

CanPersonalizeWebParts: This is **true** if the current user has permission to set personalizable properties (section 3.1.1.1.1).

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

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

2.2.2.33 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;
        String[] _links;
    }
}
```

_allUserProperties: Serialized Web Part properties (section 2.2.3.1) representing 0 or more customizable properties (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 (section 3.1.1.1.1) on the Web Part.

_webPartIdProperty: The identifier of the Web Part.

_links: Property Link information of the Web Part<12>.

2.2.2.34 WebPartChromeDataSet

This class contains an array of **WebPartVerbData** (section 2.2.2.37) to display on the Web Part chrome.

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

```
}
```

WebPartVerbList: An array of **WebPartVerbData** (section 2.2.2.37) to display on the Web Part chrome.<13>

2.2.2.35 WebPartData

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

```
namespace Microsoft.SharePoint.UserCode
{
    class WebPartData
    {
        Guid StorageKey;
        string ZoneId;
        Int32 ZoneIndex;
        bool IsClosed;
        bool IsStatic;
        byte ChromeState;
        Guid WebPartTypeID;
        String AssemblyFullName;
        String TypeFullName;
        byte[] AllUsersProperties;
        byte[] PerUserProperties;
        String[] Links;
        byte WebPartType;
        byte[] AllUsersCache;
        byte[] PerUserCache;
        String Source;
        byte DataOrigin;
        String NamingContainerID;
        String NamingContainerUniqueID;
        String ID;
        String WebPartQualifierID;
        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 **IsStatic** is **true**.

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

IsStatic: 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.

WebPartTypeID: This field MUST be ignored.

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 represents 0 or more customizable properties (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 (section 3.1.1.1.1) on the Web Part.

Links: Property Link information of the Web Part<14>.

WebPartType: Indicates whether customization or personalization is in effect.

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

AllUsersCache: This field MUST be ignored.

PerUserCache: This field MUST be ignored.

Source: This field MUST be ignored.

DataOrigin: This field MUST be ignored.

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

ID: Identifier of the Web Part. This field MUST be ignored by the protocol server if **IsStatic** is **true**.

WebPartQualifierID: This field MUST be ignored.

EnablePagePropertiesForWebPart: 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.

2.2.2.36 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 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 rendering if **WebPartData.IsStatic** (section 2.2.2.36) is **false**. The unique identifier MUST be generated by prepending **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**. 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. This object is serialized according to [MS-NRTP] section 2.2.2.1.

FormID: Identifier of the **Form** element that contains the Web Part.

FormUniqueID: Unique identifier of the **Form** element that contains the Web Part. The **FormUniqueID** can be used to generate an identifier to the **Form** element to be used 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 2.2.2.36) is **true**.

ZoneAllowPersonalization: This is **true** if Web Parts within the Web Part zone are allowed to update personalizable properties (section 3.1.1.1.1); **false** otherwise. The protocol server MUST ignore this field if **WebPartData.IsStatic** (section 2.2.2.36) is **true**.

ZoneAllowCustomization: This is **true** if Web Parts within the Web Part zone are allowed to update customizable properties (section 3.1.1.1.1); **false** otherwise. The protocol server MUST ignore this field if **WebPartData.IsStatic** (section 2.2.2.36) 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 2.2.2.36) 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 2.2.2.36) 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 2.2.2.36) is **true**.

ZoneLayoutOrientation: Orientation of Web Parts within the Web Part zone. The protocol server MUST ignore this field if **WebPartData.IsStatic** (section 2.2.2.36) is **true**. The following table described possible values.

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 2.2.2.36) 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 2.2.2.36) 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 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.

ToolPaneParentID: Identifier of the Web Part toolpane parent.

ToolPaneID: Identifier of the Web Part toolpane.

2.2.2.37 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-262-1999], to be invoked when the Web Part verb is executed.

Enabled: This is **true** if the Web Part verb is enabled; **false** if it is disabled. If a Web Part verb is displayed in UI a disabled Web Part verb might commonly be displayed as grayed out and an enabled Web Part verb might commonly be displayed as not grayed out.

Checked: This is **true** if the Web Part verb is checked; **false** if it is not checked. If a Web Part verb is displayed in UI a checked Web Part verb might commonly be displayed with a check mark next to it and a Web Part verb that is not checked might be commonly displayed without a check mark next to it.

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

2.2.2.38 WorkerRequestData

This class contains HTTP data for the Web Part request.

```
namespace Microsoft.SharePoint.UserCode
{
    class WorkerRequestData
    {
        String[][2]  serverVariables;
        String[][] _unknownHeaders;
        String[] _knownHeaders;
        byte[]  inputStream;
        String _filePath;
        String  queryString;
        String _rawUrl;
        String _pathInfo;
    }
}
```

_serverVariables: Collection of name-value pairs that provide information about the protocol client's front-end Web server and its current page request.<15>

_unknownHeaders: List of non-standard HTTP Header values in a two dimensional string array. The **Array** type is defined in [MS-NRTP]. The **Array** elements are of string array format (see [MS-NRTP] section 3.1.5.1.7).

_knownHeaders: Standard HTTP headers that correspond to the specified indexes, in string array format (see [MS-NRTP] section 3.1.5.1.7).

Index	HTTP header
0	**Cache-Control**
1	**Connection**
2	**Date**
3	**Keep-Alive**
4	**Pragma**
5	Trailer

Index	HTTP header
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
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

Index	HTTP header
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: MUST be ignored.

_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**.

2.2.3 Complex Types

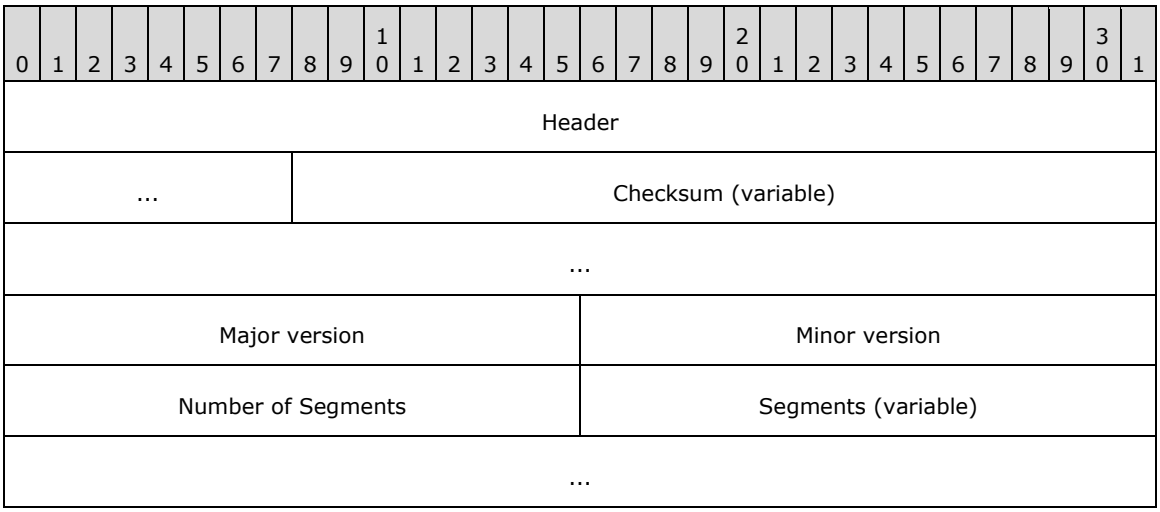
This section specifies complex types passed between the protocol client and protocol server.

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.



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.

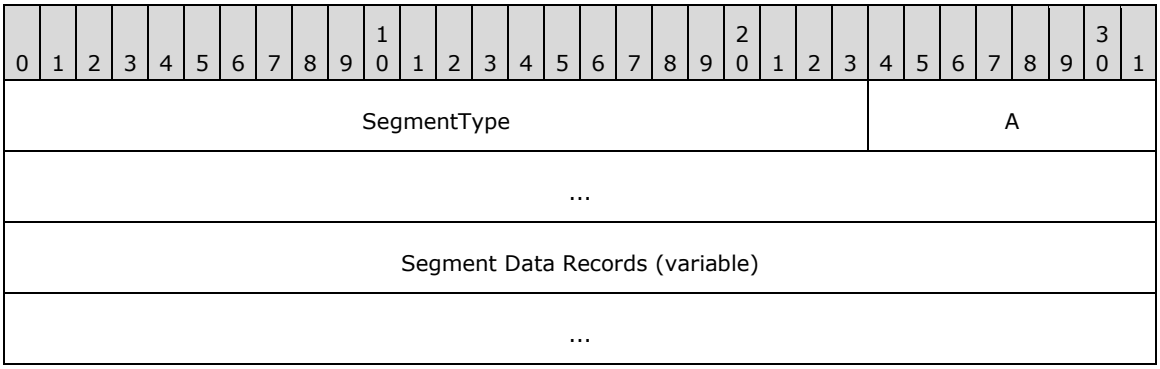
Minor version (2 bytes): A 16-bit unsigned integer. This MUST be 0x0203.

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

A **Segment Record** describes the different types of data that can be stored in a segment.



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

A - 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 start 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.

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**.

2.2.3.1.3 Format of Property Name Record

The **Property Name** record is used to store the name of a property.

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

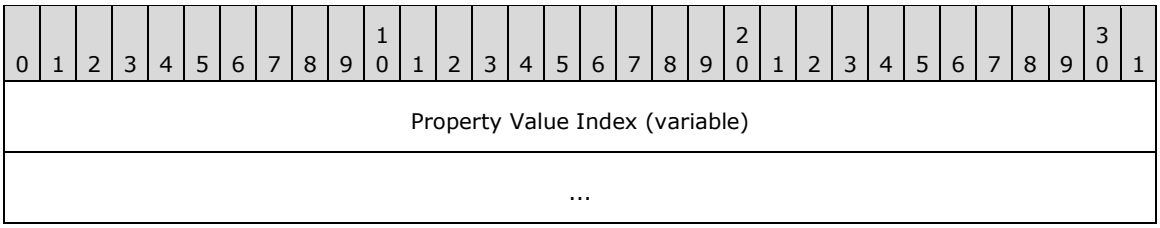
2.2.3.1.4 Format of Property Value Record

The **Property Value** record is used to store the value of a property.

The records are stored as **Property Value** structures (section 2.2.3.1.5).

2.2.3.1.5 Format of the Property Value Index Record

The **Property Value Index** is used to reference a **Property Value** record. It has the following format.



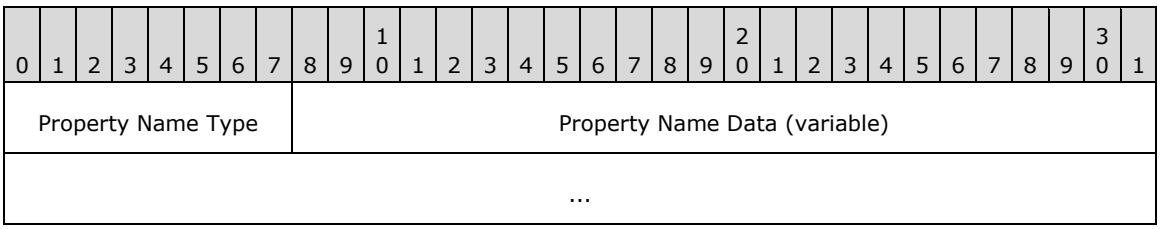
Property Value Index (variable): Index of the **Property Value Record**. The **Property Value index** is stored as a **7BitEncodedInt** (section 2.2.3.1.8).

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

1. Start with a count of 3.
2. Move forward from the **Header** record adding 2 to the count for each **Segment Record** (section 2.2.3.1.2) and 1 for each **Segment Data Record** encountered until 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. They have the following format.



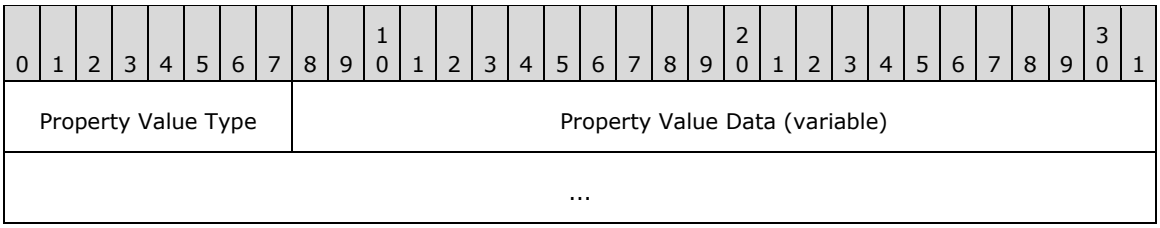
Property Name Type (1 byte): Identifies the format of the **Property Name Data**.

Property Name Data (variable): The **Property Name Data** is in the format specified by the **Property Name Type Field**, as described in the following table.

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. They have the following format.



Property Value Type (1 byte): Indicates the **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, as described in the following table.

Property Value Type	Property Value Data
0x01	**Int16 (2 bytes):** Data is written least significant byte 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).
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) ▪ Alpha is set to `0xFF & (value >> 0x18);` ▪ Red is set to `0xFF & (value >> 0x10);` ▪ Green is set to `0xFF & (value >> 0x08);` ▪ Blue is set to `0xFF;`
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)

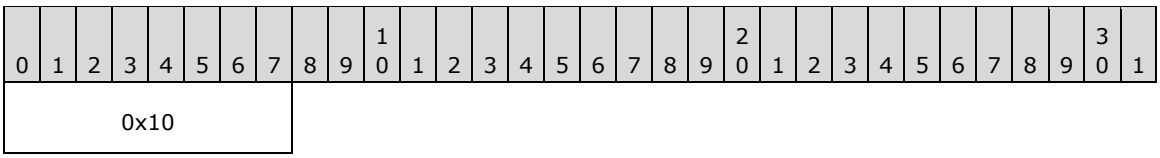
Property Value Type	Property Value Data
0x66	**ZeroInt32** (0 bytes)
0x67	Boolean **true** (0 bytes)
0x68	Boolean **false** (0 bytes)

2.2.3.1.8 7BitEncodedInt Structure

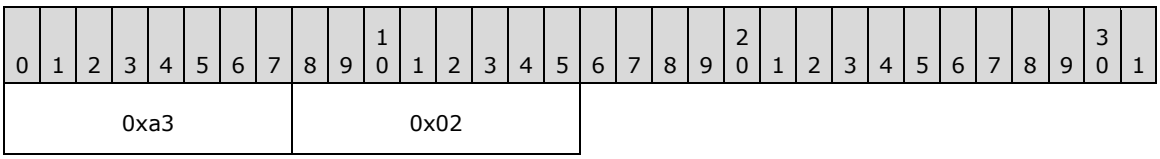
To save space when persisting integer values the **7bitEncodedInt** structure is used.

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 0x10 would be stored as follows.

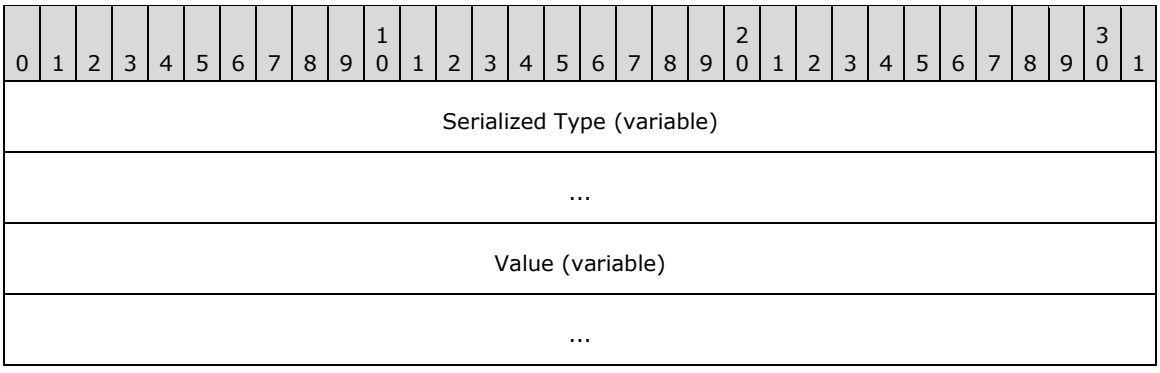


For a larger value such as 0x123 it is written as follows.



2.2.3.1.9 IntEnum Structure

Structure used for writing out an enumeration. This structure has the following format.

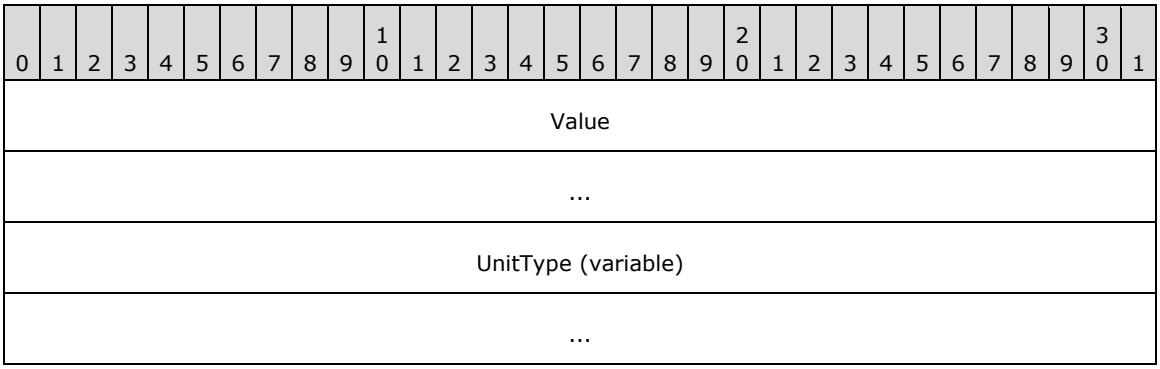


Serialized Type (variable): The type information for the enum stored as a **Serialized Type Structure** (section 2.2.3.1.12).

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

2.2.3.1.10 Unit Structure

Structure used to represent a length of measurement. This structure has the following format.



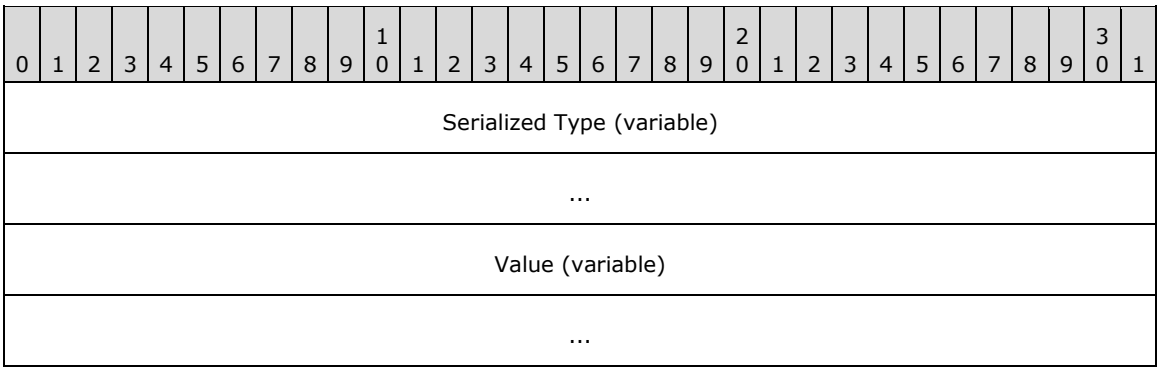
Value (8 bytes): Value of the unit in double-precision 64 bit [IEEE754] format.

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

Unit Type	Unit
0x01	Pixel
0x02	Point
0x03	Pica
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. This structure has the following format.

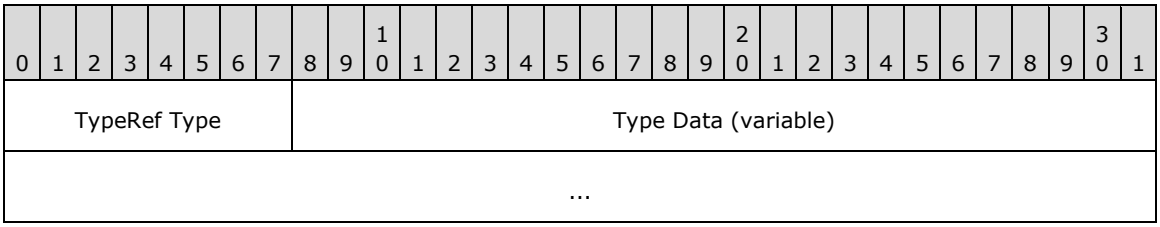


Serialized Type (variable): Type information for the property in **Serialized Type Structure** (section 2.2.3.1.12).

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. This structure has the following format.



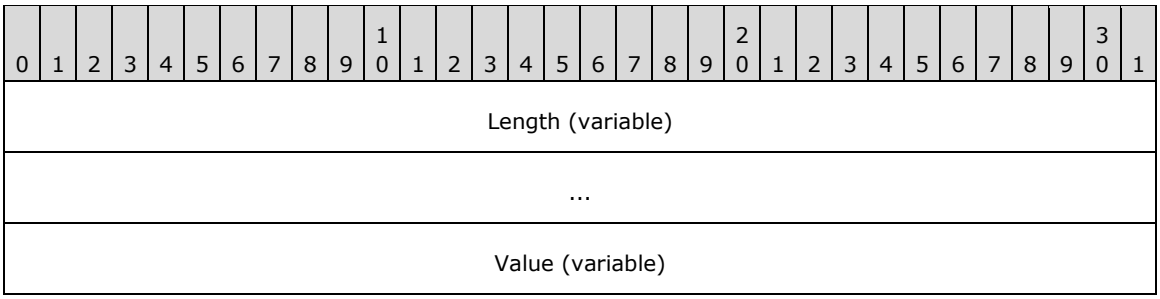
TypeRef Type (1 byte): Indicates how to decode the Type Data.

Type Data (variable): Data that represents the type. Data is in the format specified by the **TypeRef Type** field. Possible values are described in the following table.

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 ▪ **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. This structure has the following format.



...

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. Possible values are described in the following table.

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
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

Token	String
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
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

Token	String
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
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

Token	String
0x00000064	NamespaceMembersWebPart
0x00000065	CacheDataStorage
0x00000066	CacheDataTimeout
0x00000067	CacheXslStorage
0x00000068	AlternativeText
0x00000069	DataSourceBindings
0x0000006A	Template
0x0000006B	NamespaceWebPartV3
0x0000006C	ID
0x0000006D	AttachedPropertiesShared
0x0000006E	AttachedPropertiesUser
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

Token	String
0x00000085	Direction
0x00000086	ExportMode
0x00000087	HelpUrl
0x00000088	Hidden
0x00000089	ImportErrorMessage
0x0000008A	IsClosed
0x0000008B	TitleIconImageUrl
0x0000008C	ZoneIndex
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. Possible values are described in the following table.

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

Token	Color name
0x23	Black
0x24	BlanchedAlmond
0x25	Blue
0x26	BlueViolet
0x27	Brown
0x28	BurlyWood
0xA8	ButtonFace
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

Token	Color name
0x3B	DarkRed
0x3C	DarkSalmon
0x3D	DarkSeaGreen
0x3E	DarkSlateBlue
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
0x0C	GrayText
0x4F	Green
0x50	GreenYellow
0x0D	Highlight
0x0E	HighlightText
0x51	Honeydew
0x52	HotPink
0x0F	HotTrack
0x10	InactiveBorder
0x11	InactiveCaption

Token	Color name
0x12	InactiveCaptionText
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
0x6F	MediumOrchid
0x70	MediumPurple

Token	Color name
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
0x89	Pink
0x8A	Plum
0x8B	PowderBlue
0x8C	Purple
0x8D	Red

Token	Color name
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
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

This section describes a conceptual model of possible data organization which an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model, as long as their external behavior is consistent with the behavior described in this document.

The back-end database server maintains the following sets of data for this protocol within both a **configuration database** and one or more **content databases**. Data within the databases is maintained until updated or removed.

3.1.1.1 Web Parts

3.1.1.1.1 Customizable and Personalizable Properties

A Web Part defines a number of properties that can be modified to change how the Web Part behaves or renders. The properties are split into two groups, customizable and personalizable. These two groups of properties are stored in the back end database server for each Web Part, and both sets of properties are used to instantiate and render a Web Part on a front-end Web server. It is up to the Web Part implementer to determine whether a property is customizable or personalizable. A property is customizable if all users accessing the Web Part MUST get the same value for the property. A property is personalizable if users accessing the Web Part MUST be able to modify the property to a value specific to each user.

3.1.1.1.2 Adding and Modifying a Web Part for All Users (Customization)

When a Web Part is added to the shared view of a Web Parts page a new entry for the Web Part is added into the back end database server containing all the personalizable and customizable properties of the Web Part. For each version of a Web Parts page there is only one copy of the personalizable and customizable properties stored in the back end database server for the shared view of a Web Part. As a result, when two different users browse to the shared view of the same Web Parts page the same set of personalizable and customizable properties for the Web Part are returned, resulting in the same Web Part being rendered for each user. Modifying this copy of properties used to render the shared view of a Web Part is called customization, and all users browsing to the shared view of the Web Parts page will see the same customized Web Part.

3.1.1.1.3 Adding a Web Part for All Users then modifying it uniquely for a particular User (Personalization)

When a Web Part is added to the shared view of a Web Parts page and a user then accesses the shared view or personal view of the Web Parts page, the personalizable and customizable properties returned for the Web Part will be the same so the Web Part will render the same in both the shared view and personal view.

If the user then modifies the Web Part from the personal view of the Web Parts page, then all of the personalizable properties currently stored in the back end database server for the Web Part are copied into a separate entry in the back end database server for the Web Part that is associated with the particular user who modified the Web Part.

This process is called personalization, and it means there are now two copies of the personalizable properties for the Web Part in the back end database server, one copy that is used when any user accesses the Web Part in the shared view of the Web Parts page or they access the Web Part in the personal view of the Web Parts page but have not yet personalized the Web Part, and a second copy that is used when the user who personalized the Web Part accesses the Web Part in the personal view of the Web Parts page.

Every time a different user personalizes the Web Part an additional copy of the personalizable properties are stored for the Web Part in the back end database server for that particular user. When a user accesses the personal view of a Web Parts page, personalizable and customizable properties for the Web Part will be returned. If the Web Part has not been personalized by this user then these properties will be the same ones that are returned if the user browsed to the shared view of the Web Parts page. If the Web Part has been personalized by this user then the personalizable properties will be a unique copy that is stored in the back end database server just for this user, the customizable properties will be the same ones that are returned when accessing the shared view of the Web Parts page. There is only one copy of the customizable properties of a Web Part for a particular version of a Web Parts page, there is one copy of the personalizable properties of a Web Part for each user who has personalized that Web Part on the Web Parts page.

3.1.1.1.4 Adding a Web Part just for a particular User (Personal Web Part)

When a Web Part is added to the personal view of a Web Parts page a new entry for the Web Part is added into the back end database server containing all the personalizable and customizable properties of the Web Part, and the entry is associated with the particular user who added the Web Part. This is called a **personal Web Part** and it will only be returned when the user who added the Web Part is accessing the Web Parts page in personal view. No one else will ever have access to this personal Web Part. If a personal Web Part is modified the one copy of the personalizable and customizable properties for the Web Part in the back end database server will be updated, and again only the user who added the personal Web Part will see the changes when they access the personal view of the Web Parts page.

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.
InitializeLifetimeService	This operation is reserved and MUST NOT be called.

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 can contain the type of any object.<16>

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

userToken: A token (section 2.2.2.30) 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: If an error occurs while processing this method, one of the exceptions that are listed in the following table will be raised.

Exception	Description
SPUserCodeExecutionPipelineTerminallyFailedException (section 2.2.2.14)	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 2.2.2.15)	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 2.2.2.18)	The protocol server MUST throw this exception when the user code being executed on the protocol server has failed.
SPUserCodeValidationFailedException (section 2.2.2.19)	The protocol server MUST throw this exception when a failure is encountered while validating the assembly group that contains the user code.
Any exception raised 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: No exceptions are thrown beyond those raised 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 SAO 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 remoting server is a Single-Call SAO at "tcp://RemotingMachine: portNumber". Where the portNumber is the port the protocol client and protocol server have agreed to use for their communicating with each other.

The remotely running **Ping** method returns **true** in the response message to the client.

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

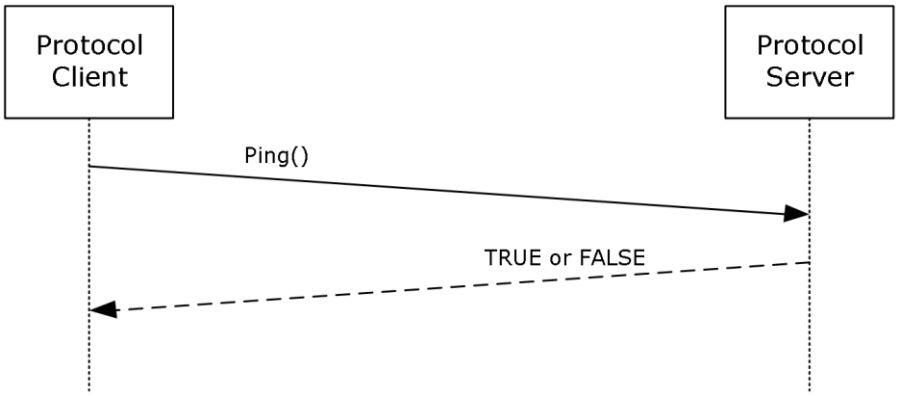


Figure 3: Sequence diagram for this message exchange pattern

A dump 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 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:               (00000000000000000..............)
    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 computers 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
- Microsoft SharePoint Foundation 2013
- Microsoft SharePoint Server 2016

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 2.2.1.8: This enumeration is available only in SharePoint Foundation 2013.

<2> Section 2.2.2.1: In SharePoint Foundation 2010 the **changeProperties** field of **EventResults** is a **System.Collections.Hashtable**.

<3> Section 2.2.2.3: This class is available only in SharePoint Foundation 2013.

<4> Section 2.2.2.4: This class is available only in SharePoint Foundation 2013.

<5> Section 2.2.2.5: This class is available only in SharePoint Foundation 2013.

<6> Section 2.2.2.9: In SharePoint Foundation 2010 the **m_properties** field of **SPItemEventDataCollection** is a **System.Collections.Hashtable**.

<7> Section 2.2.2.9: In SharePoint Foundation 2010 the **m_changeProperties** field of **SPItemEventDataCollection** is a **System.Collections.Hashtable**.

<8> Section 2.2.2.13: In SharePoint Foundation 2010 the **solutionValidationErrorUrl** field of **SPUserCodeExecutionContext** will not be present.

<9> Section 2.2.2.13: In SharePoint Foundation 2010 the **solutionValidationErrorMessage** field of **SPUserCodeExecutionContext** will not be present.

<10> Section 2.2.2.16: In SharePoint Foundation 2010 the **Parameters** field on **SPUserCodeFeatureCallOutContext** is a **System.Collections.Generic.Dictionary<string, string>**.

<11> Section 2.2.2.28: In SharePoint Foundation 2010 the **Parameters** field on **SPUserCodeWorkflowActionSandboxExecutionContext** is a **System.Collections.Hashtable**.

<12> Section 2.2.2.33: This field will be removed in Microsoft SharePoint Foundation 2010.

<13> Section 2.2.2.34: In SharePoint Foundation 2010 the **WebPartVerbList** field of **WebPartChromeDataSet** is a **System.Collections.ArrayList**.

<14> Section 2.2.2.35: This field will be removed in SharePoint Foundation 2010.

<15> Section 2.2.2.38: In SharePoint Foundation 2010 the **serverVariables** field on **WorkerRequestData** is a **System.Collections.Specialized.NameValueCollection**.

<16> Section 3.1.5.1: In SharePoint Foundation 2010 there is a relationship between the type of object passed to the *userCodeWrapperType* parameter of the **Execute** method and the type of object that is passed to the *executionContext* parameter, 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

This section identifies changes that were made to this document since the last release. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- The removal of a document from the documentation set.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.
- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated**.

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
6 Appendix A: Product Behavior	Updated list of supported products.	Y	Content updated due to protocol revision.

8 Index

N

O

P

R

S

T

U

V

W