[MS-PPSSPROC]: PerformancePoint Services Stored Procedures Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft 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.
- **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.

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release (beta) version of this technology. This Open Specification is final documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release (beta) versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Revision History	Revision Class	Comments
07/13/2009	0.1	Major	Initial Availability
08/28/2009	0.2	Editorial	Revised and edited the technical content
11/06/2009	0.3	Editorial	Revised and edited the technical content
02/19/2010	1.0	Editorial	Revised and edited the technical content
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Minor	Updated the technical content
06/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	1.5	Minor	Clarified the meaning of the technical content.
04/11/2012	1.5	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	1.5	No change	No changes to the meaning, language, or formatting of

Date	Revision History	Revision Class	Comments
			the technical content.

Table of Contents

1 Introduction

This document specifies the PerformancePoint Services Stored Procedures Protocol, which defines communication requests for scorecard annotations administration, user-selected filters for user administration, and temporary state data structures for first-class object administration tasks between the front-end Web server and the database server. This server-to-server protocol uses the PerformancePoint Services Application Server Protocol.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

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

serialize

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

back-end database server
dashboard
data source
filter
first-class object
front-end Web server
hash code
key performance indicator (KPI)
report view
result set
return code
scorecard
service application
site
slice
stored procedure
Structured Query Language (SQL)
table
Transact-Structured Query Language (T-SQL)
view
Web application

The following terms are specific to this document:

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

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

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

[Iseminger] Microsoft Corporation, "SQL Server 2000 Architecture and XML/Internet Support", Volume 1 of Microsoft SQL Server 2000 Reference Library, Microsoft Press, 2001, ISBN 0-7356-1280-3, http://www.microsoft.com/mspress/books/5001.aspx

[MSDN-TSQL-Ref] Microsoft Corporation, "Transact-SQL Reference", http://msdn.microsoft.com/en-us/library/ms189826(SQL.90).aspx

[MS-PPSAPP] Microsoft Corporation, "PerformancePoint Services Application Server Protocol Specification".

[MS-PPSAS] Microsoft Corporation, "PerformancePoint Services AuthoringService Protocol Specification".

[MS-TDS] Microsoft Corporation, "Tabular Data Stream Protocol Specification".

[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

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

1.2.2 Informative References

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

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

1.3 Protocol Overview (Synopsis)

Business intelligence platforms typically support workflow whereby a user creates and publishes content representing business performance indicators, reports in a variety of forms, and **data source (1)** references. There are certain types of data stored in the back-end database server which are required to perform some business intelligence tasks.

This protocol specifies the communication requests between the **front-end Web server** and the **back-end database server** to perform these tasks: maintain scorecard annotations, pass user selected filters between PerformancePoint **first-class objects** in a dashboard and connect a filter to an analytic grid/chart or a scorecard, and maintain state information when navigating among PerformancePoint first-class objects.

This server-to-server protocol uses the PerformancePoint Services Application Server Protocol, as described in [MS-PPSAPP], as its transport between the front-end Web server and the back-end database server.

This protocol enables a protocol client to do the following:

- Add, change, retrieve and delete scorecard annotations from the database on the back-end database server.

- Add, retrieve, and delete user selected filters for a user from the database on the back-end database server.
- Add, retrieve, and delete temporary copies of PerformancePoint first-class object from the database on the back-end database server.

A typical use of this protocol is a service application that is called by one or more other **service applications** allowing users to annotate or make comments on scorecards rendered by a Web application (2).

Another typical use of this protocol is a service application that is called by one or more other service applications, is to pass user selected filters between PerformancePoint first-class objects in a dashboard and connect a filter to an analytic grid/chart or a scorecard.

Another typical use of this protocol is a service application that is called by one or more other service applications, is to maintain state information when navigating among PerformancePoint first-class objects for a certain period of time, and deleting the state information after the time expires.

1.3.1 Annotations

The protocol allows clients to add, change, retrieve, and delete **scorecard** annotations on the back-end database server. The following diagram specifies the data flow between the protocol client and the protocol server with regards to administering scorecard annotations. For more details on the stored procedures in this diagram, see section 3.1.5.

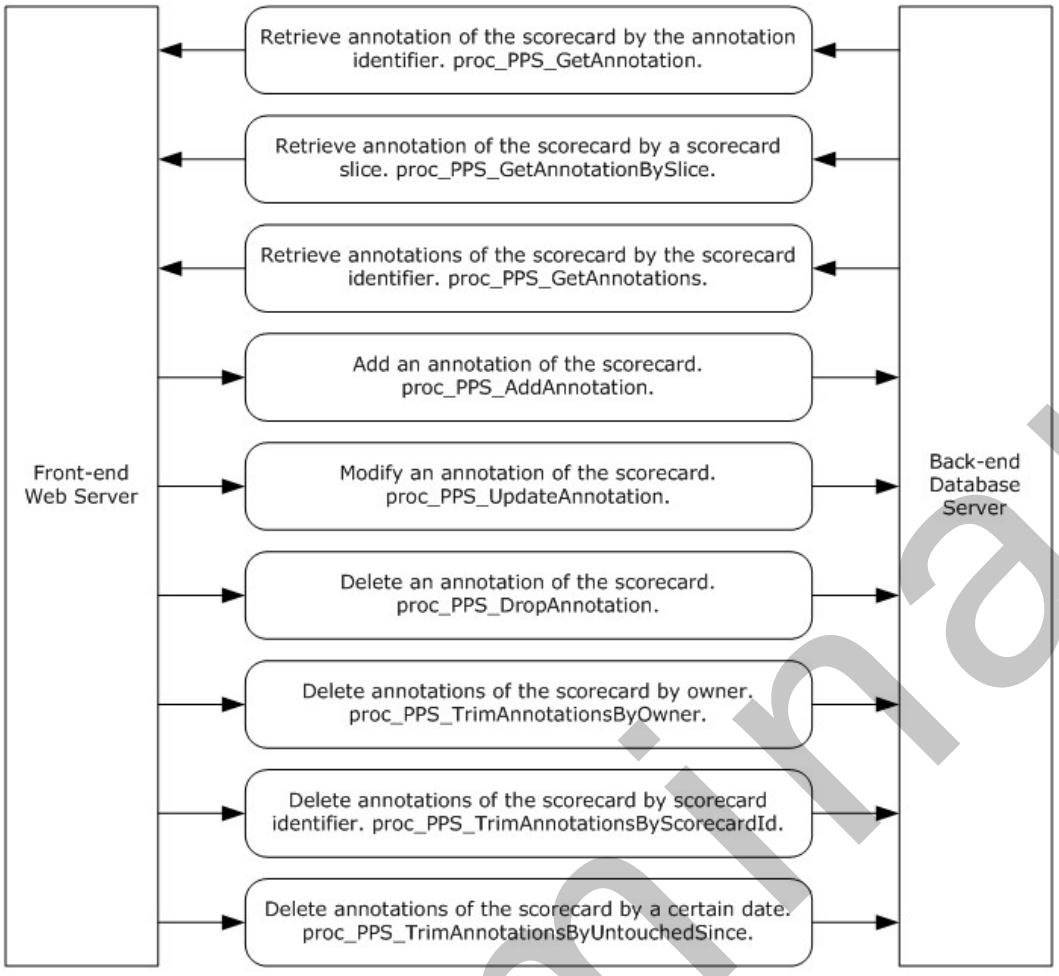

Figure 1: Scorecard Annotation Administration data-flow diagram

The protocol allows clients to retrieve an annotation, based on an annotation identifier, or to retrieve annotations of the scorecard based on scorecard identifier or on a scorecard slice. The client can modify the annotation of the scorecard. The client can also delete an annotation by annotation identifier, scorecard identifier, or scorecard owner, or delete annotations which have not been touched since a specific date.

1.3.2 ParameterValues

The protocol allows clients to add, change, retrieve, and delete user selected filters for a user on the back-end database server. The relational table to store these data is called **PPSParameterValues** table. The records in **PPSParameterValues** table are called **ParameterValues** records here.

The following diagram specifies the data flow between the protocol client and the protocol server with regards to administration of the user selected filters for a user.

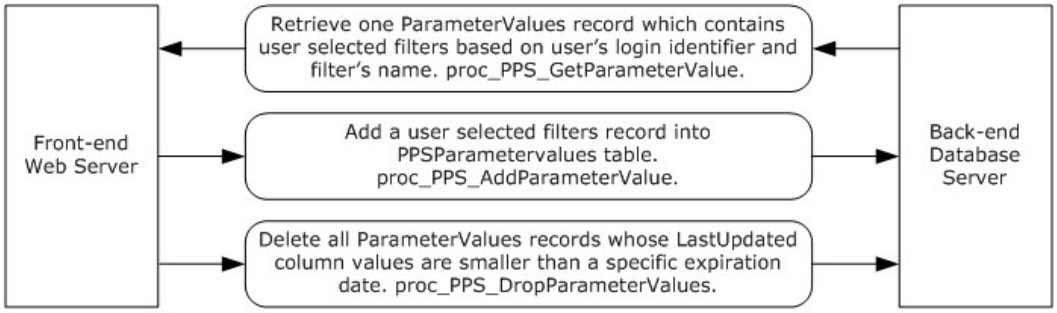

Figure 2: User Selected Filters Administration data-flow diagram

The protocol allows clients to retrieve one **ParameterValues** record which contains user selected filters based on user's login identifier and filter's name. The client can insert a user selected filters record into **PPSParameterValues** table (see section 2.2.5.2). The client can also delete all **ParameterValues** records which have not been touched since a specific date.

1.3.3 TempFCOs

The protocol allows clients to add, change, retrieve, and delete the temporary state data structures for PerformancePoint first-class object on the back-end database server. The relational table to store these data is called the **PPSTempFCOs** table. The records in the **PPSTempFCOs** table are called **TempFCO** records here.

The following diagram specifies the data flow between the protocol client and the protocol server with regards to administration of the PerformancePoint Server **TempFCO** records.

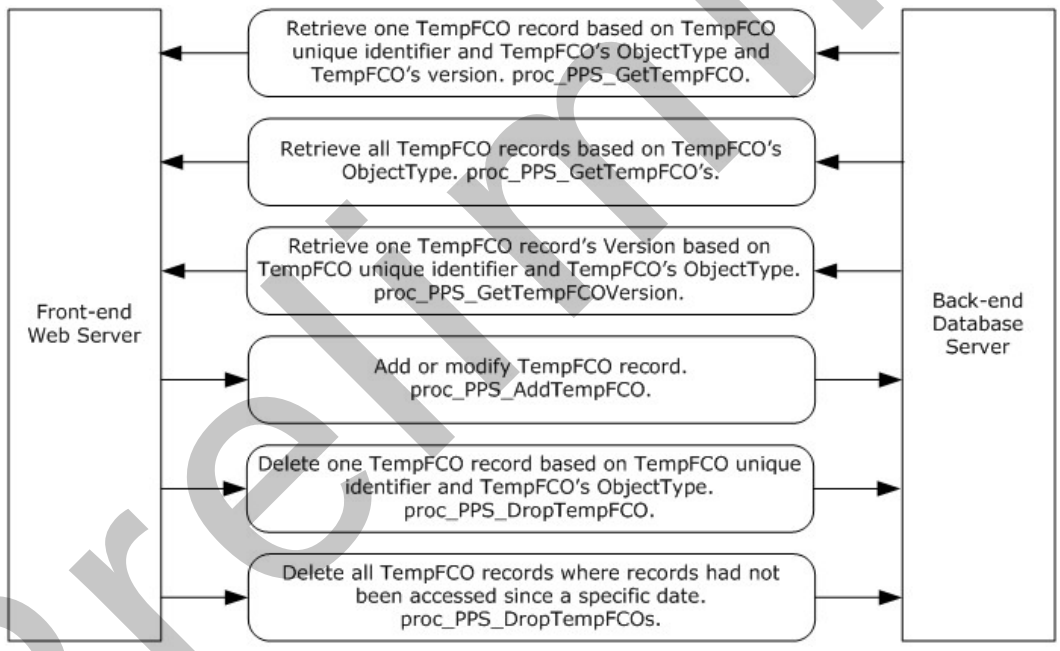

Figure 3: TempFCO Records Administration data-flow diagram

The protocol allows clients to retrieve all **TempFCO** records based on **TempFCO** record's **ObjectType** value, or to retrieve one **TempFCO** record's version based on **TempFCO** record

identifier and its **ObjectType** value. The client can insert a **TempFCO** record into **PPSTempFCOs** table or update an existing **TempFCO** record. The client can also delete one **TempFCO** record based on **TempFCO** record identifier and **ObjectType** value, or delete all **TempFCO** records which have not been touched since a specific date.

1.4 Relationship to Other Protocols

This protocol relies on the PerformancePoint Services Application Server Protocol, as described in [MS-PPSAPP], as its transport protocol to call **stored procedures** to manipulate and query scorecard annotations, user selected filters, and temporary copies of PerformancePoint first-class object via **result sets** and **return codes**.

This relationship is illustrated in the following diagram:

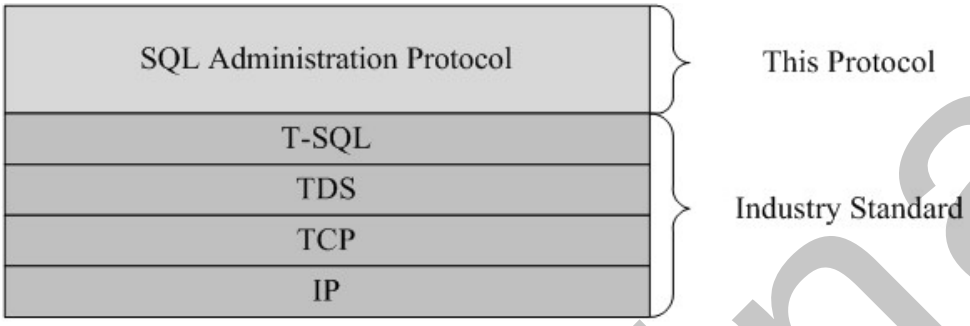

Figure 4: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

Unless otherwise specified, the stored procedures and any related tables are present in the database that is being queried on the back-end database server. The tables in the database contain valid data in a consistent state to be queried successfully by the stored procedures.

The user that calls the stored procedures has adequate permission to access the databases that contain the stored procedures.

1.6 Applicability Statement

This protocol is only applicable to front-end Web servers when communicating with the back-end database server to satisfy requests for tasks such as manipulating annotations of scorecards, user selected filters, and temporary copies of PerformancePoint first-class object.

1.7 Versioning and Capability Negotiation

Version negotiation: Versions of the data structures or stored procedures in the database require the same calling parameters and return code values that are expected by the protocol client in order for the stored procedures to be called correctly. If the stored procedures are not provided to the expected calling parameters or return code values, the results of the call are indeterminate.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

[MS-TDS] is the transport protocol used to call the stored procedures, query **SQL Views** or SQL **Tables** and return result codes and result sets.

2.2 Common Data Types

This section contains common definitions used by this protocol.

2.2.1 Simple Data Types and Enumerations

2.2.1.1 Object Type

A unique identifier of an object type. Valid values are listed in the following table.

Value	Description
1	KPI [MS-PPSAS]
2	Scorecard [MS-PPSAS]
3	**Report view** [MS-PPSAS]
4	Indicator [MS-PPSAS]
5	Data source(1) [MS-PPSAS]
6	**Dashboard** [MS-PPSAS]
7	Temporary report view [MS-PPSAS]
8	Filter [MS-PPSAS]

2.2.2 Bit Fields and Flag Structures

None.

2.2.3 Binary Structures

None.

2.2.4 Result Sets

The following shows all the result sets returned from the stored procedures listed in this document.

2.2.4.1 proc_PPS_GetAnnotation.ResultSet0

This result set is used by the **proc_PPS_GetAnnotation** stored procedure. It contains information about cells' annotations in a scorecard with eleven columns returned. The result set MUST contain zero or more rows, each corresponding to a single annotation. The **T-SQL** syntax for the result set is as follows:

```
ScorecardId uniqueidentifier,
```

```
ConfiguredViewId uniqueidentifier,
SliceHashCode int,
Slice ntext,
CellPath ntext,
CellPathHashCode int,
Created datetime,
Owner nvarchar(1000),
LastUpdated datetime,
LastUpdatedBy nvarchar(1000),
SerializedXml ntext,
```

ScorecardId: The unique identifier of a scorecard. The value MUST NOT be NULL.

ConfiguredViewId: The unique identifier of a configured view. Configured view is the view definition of the scorecard. There is only one configured view per scorecard. The value MUST NOT be NULL.

SliceHashCode: The **hash code** of the **slice** within a scorecard. The slice hash code is used to retrieve an associated annotation. The value MUST NOT be NULL.

Slice: The slice data. It is used in conjunction with column **CellPath** to identify a single cell in a grid. The value MUST NOT be NULL.

CellPath: The path to the cell data. It is used in conjunction with column Slice to identify a single cell in a grid. The value MUST NOT be NULL.

CellPathHashCode: The hash code of the cell path. The cell path hash code is used to retrieve an associated annotation. The value MUST NOT be NULL.

Created: The date and time of this annotation's creation. This value MUST NOT be NULL.

Owner: The login of the user who created this annotation. This value MUST NOT be NULL.

LastUpdated: The date and time of this annotation's latest modification. This value MUST NOT be NULL.

LastUpdatedBy: The login of the user who last changed this annotation. This value MUST NOT be NULL.

SerializedXml: The XML [XMLNS] serialization (see **serialize**) of an annotation object for a cell's annotation in a scorecard. This value MUST NOT be NULL.

2.2.4.2 proc_PPS_GetAnnotationBySlice.ResultSet0

This result set is used by the **proc_PPS_GetAnnotationBySlice** stored procedure. It contains information about cells' annotations with nine columns returned. The result set MUST contain zero or more rows, each corresponding to a single annotation. The T-SQL syntax for the result set is as follows:

```
AnnotationId uniqueidentifier,
ConfiguredViewId uniqueidentifier,
Slice ntext,
CellPath ntext,
Created datetime,
Owner nvarchar(1000),
LastUpdated datetime,
```

```
LastUpdatedBy nvarchar(1000),
SerializedXml ntext,
```

AnnotationId: The unique identifier of an annotation. The value MUST NOT be NULL.

ConfiguredViewId: The unique identifier of configured view. Configured view is the view definition of the scorecard. There is only one configured view per scorecard. The value MUST NOT be NULL.

Slice: The slice data. It is used in conjunction with column CellPath to identify a single cell in a grid. The value MUST NOT be NULL.

CellPath: The cell path data. It is used in conjunction with column Slice to identify a single cell in a grid. The value MUST NOT be NULL.

Created: The date and time of this annotation's creation. This value MUST NOT be NULL.

Owner: The login of the user who created this annotation. This value MUST NOT be NULL.

LastUpdated: The date and time of this annotation's latest modification. This value MUST NOT be NULL.

LastUpdatedBy: The login of the user who last changed this annotation. This value MUST NOT be NULL.

SerializedXml: The XML serialization of an annotation object for a cell’s annotation in a scorecard. This value MUST NOT be NULL.

2.2.4.3 proc_PPS_GetAnnotations.ResultSet0

This result set is used by the **proc_PPS_GetAnnotations** stored procedure. It contains information about cells' annotations with nine columns data returned. The result set MUST contain zero or more rows, each corresponding to a single annotation. The T-SQL syntax for the result set is as follows:

```
AnnotationId uniqueidentifier,
ConfiguredViewId uniqueidentifier,
Slice ntext,
CellPath ntext,
Created datetime,
Owner nvarchar(1000),
LastUpdated datetime,
LastUpdatedBy nvarchar(1000),
SerializedXml ntext,
```

AnnotationId: The unique identifier of an annotation. The value MUST NOT be NULL.

ConfiguredViewId: The unique identifier of a configured view. Configured view is the view definition of the scorecard. There is only one configured view per scorecard. The value MUST NOT be NULL.

Slice: The slice data. It is used in conjunction with column **CellPath** to identify a single cell in a grid. The value MUST NOT be NULL.

CellPath: The cell path data. It is used in conjunction with column Slice to identify a single cell in a grid. The value MUST NOT be NULL.

Created: The date and time of this annotation's creation. This value MUST NOT be NULL.

Owner: The login of the user who created this annotation. This value MUST NOT be NULL.

LastUpdated: The date and time of this annotation's latest modification. This value MUST NOT be NULL.

LastUpdatedBy: The login of the user who last changed this annotation. This value MUST NOT be NULL.

SerializedXml: The XML serialization of an annotation object for a cell's annotation in a scorecard. This value MUST NOT be NULL.

2.2.4.4 proc_PPS_GetParameterValue.ResultSet0

This result set is used by the **proc_PPS_GetParameterValue** stored procedure. It contains information about user selected filters for a user with one column (**SerializedXml**) data returned. The result set MUST contain zero or more rows, each corresponding to a single user selected filter record. The T-SQL syntax for the result set is as follows:

```
SerializedXml ntext,
```

SerializedXml: The XML serialization of the single column data table which contains user selected filters for the user login. This value MUST NOT be NULL.

2.2.4.5 proc_PPS_GetTempFCO.ResultSet0

This result set is used by the **proc_PPS_GetTempFCO** stored procedure. It contains information about **TempFCO** records with three columns (**ElementId**, **Version**, **SerializedXml**) data returned. **TempFCO** is the temporary copies of PerformancePoint first-class object. The result set MUST contain zero or more rows, each corresponding to a single **TempFCO** record. The T-SQL syntax for the result set is as follows:

```
ElementId uniqueidentifier,
Version int,
SerializedXml ntext,
```

ElementId: The unique identifier of **TempFCO** element. **TempFCO** is the temporary copies of PerformancePoint first-class object. The value MUST NOT be NULL.

Version: The version of the **TempFCO** record. This value MUST NOT be NULL.

SerializedXml: The XML serialization of a PerformancePoint first-class object element for **TempFCO** element. The value MUST NOT be NULL.

2.2.4.6 proc_PPS_GetTempFCOs.ResultSet0

This result set is used by the **proc_PPS_GetTempFCOs** stored procedure. It contains information about **TempFCO** records with two columns data (**ElementId** and **Version**) returned. **TempFCO** is the temporary copies of PerformancePoint first-class object. The result set MUST contain zero or more rows, each corresponding to a single **TempFCO** subset. The T-SQL syntax for the result set is as follows:

```
ElementId uniqueidentifier,
Version int,
```

ElementId: The unique identifier of **TempFCO** element. **TempFCO** is the temporary copies of PerformancePoint first-class object. The value MUST NOT be NULL.

Version: The version of the **TempFCO** record. This value MUST NOT be NULL.

2.2.5 Tables and Views

The following shows all the tables that are accessed by the stored procedures listed in this document.

2.2.5.1 PPSAnnotations

The **PPSAnnotations** table contains annotation data that is tied to a cell in a scorecard.

```
ScorecardId uniqueidentifier NOT NULL,
AnnotationId uniqueidentifier NOT NULL,
ConfiguredViewId uniqueidentifier NOT NULL,
SliceHashCode int NOT NULL,
Slice ntext NOT NULL,
CellPath ntext NOT NULL,
CellPathHashCode int NOT NULL,
Created datetime NOT NULL,
Owner nvarchar(2000) NOT NULL,
LastUpdated datetime NOT NULL,
LastUpdatedBy nvarchar(2000) NOT NULL,
SerializedXml xml NOT NULL,
```

ScorecardId: The unique identifier of the scorecard. The value MUST NOT be NULL.

AnnotationId: The unique identifier of an annotation. The value MUST NOT be NULL.

ConfiguredViewId: The unique identifier of a configured view. Configured view is the view definition of the scorecard. There is only one configured view per scorecard. The value MUST NOT be NULL.

SliceHashCode: The hash code of the slice within scorecard. The slice hash code is used to retrieve associated annotation. The value MUST NOT be NULL.

Slice: The slice data. It is used in conjunction with column **CellPath** to identify a single cell in a grid. The value MUST NOT be NULL.

CellPath: The path to the cell data. It is used in conjunction with column Slice to identify a single cell in a grid. The value MUST NOT be NULL.

CellPathHashCode: The hash code of the cell path. The cell path hash code is used to retrieve associated annotation. The value MUST NOT be NULL.

Created: The date and time of this annotation's creation. This value MUST NOT be NULL.

Owner: The login of the user who created this annotation. This value MUST NOT be NULL.

LastUpdated: The date and time of this annotation's latest modification. This value MUST NOT be NULL.

LastUpdatedBy: The login of the user who last changed this annotation. This value MUST NOT be NULL.

SerializedXml: The XML serialization of an annotation object for a cell's annotation in a scorecard. This value MUST NOT be NULL.

2.2.5.2 PPSParameterValues

The **PPSParameterValues** table contains the user selected filters for an element location for a user. It is used to pass data from one PerformancePoint first-class object to another in a dashboard and connect a filter to an analytic grid/chart or the scorecard.

```
Login nvarchar(2000) NOT NULL,
ParameterUniqueName nvarchar(4096) NOT NULL,
LastUpdated datetime NOT NULL,
SerializedXml xml NOT NULL,
```

Login: The login name of the user for this user selected filters record. This value MUST NOT be NULL.

ParameterUniqueName: The name of the filters, also called filter location, it is a unique identifier for a specific connection between specific PerformancePoint first-class objects. This value MUST NOT be NULL.

LastUpdated: The date and time of this user selected filters' latest modification. This value MUST NOT be NULL.

SerializedXml: The XML serialization of the single column data table which contains user selected filters for the user login. This value MUST NOT be NULL.

2.2.5.3 PPSTempFCOs

The **PPSTempFCOs** table contains the temporary copies of PerformancePoint first-class objects. The table is used when generating analytic grid and chart views in the dashboard designer and when navigating on analytic charts and grids in either the designer or the dashboard.

```
ElementId uniqueidentifier NOT NULL,
Version int NOT NULL,
ObjectType int NOT NULL,
Name nvarchar(2000) NOT NULL,
Description nvarchar(8000) NULL,
Owner nvarchar(2000) NULL,
Created datetime NOT NULL,
LastAccessed datetime NOT NULL,
SerializedXml xml NOT NULL,
```

ElementId: The unique identifier of **TempFCO** record. **TempFCO** is the temporary copies of PerformancePoint first-class objects. This value MUST NOT be NULL.

Version: The version of the **TempFCO** record. This value MUST NOT be NULL.

ObjectType: The object type of PerformancePoint first-class object for the **TempFCO** record. The only object type which is supported for **TempFCO** record is temporary report view. This value MUST NOT be NULL. There are eight object types of PerformancePoint first-class object: KPI, scorecard, report view, indicator, data source(1), dashboard, temporary report view, and filter.

Name: The name of the **TempFCO** record. This value MUST NOT be NULL.

Description: The description of the **TempFCO** record. The value MAY be NULL.

Owner: The login of the user who created this **TempFCO** record. This value MUST NOT be NULL.

Created: The date and time of this **TempFCO** record creation. This value MUST NOT be NULL.

LastAccessed: The name of the user who last read this **TempFCO** record's **SerializedXml** column data. This value MUST NOT be NULL.

SerializedXml: The XML serialization of a PerformancePoint first-class object element for **TempFCO** element. The value MUST NOT be NULL.

2.2.6 XML Structures

No common XML Structures are defined in this protocol.

2.2.6.1 Namespaces

None.

2.2.6.2 Simple Types

None.

2.2.6.3 Complex Types

None.

2.2.6.4 Elements

None.

2.2.6.5 Attributes

None.

2.2.6.6 Groups

None.

2.2.6.7 Attribute Groups

None.

3 Protocol Details

3.1 Server Details

Server details are described in this section.

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that 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 that described in this document.

3.1.1.1 Annotations

The following table describes the abstract data model for annotations.

PPS annotations		
PK **PK** **PK**	**ScorecardId** **SliceHashCode** **CellPathHashCode**	**GUID** **INTEGER** **INTEGER**
	AnnotationId **ConfiguredViewId** **Slice** **CellPath** **Created** **Owner** **LastUpdated** **LastUpdatedBy** **SerializedXml**	**GUID** **GUID** **LONGTEXT** **LONGTEXT** **DATETIME** **VARCHAR(1000)** **DATETIME** **VARCHAR(1000)** **TEXT(0)**

Note: the column shown in bold style in the diagram indicates the column is required. Otherwise the column is not required.

The protocol server stores all implementation-specific information about annotations of a scorecard in the following sets of data:

Annotations Set: A collection of entries corresponding to annotations in a scorecard. Each entry MUST be uniquely identified by these columns' values: **ScorecardId**, **SliceHashCode**, and **CellPathHashCode**. There is no relationship between this table and other tables. Each scorecard can contain many entries of annotations. Each entry MUST include the following elements:

- **ScorecardId:** The unique identifier of the scorecard.
- **AnnotationId:** The unique identifier of an annotation.
- **ConfiguredViewId:** The unique identifier of a configured view. Configured view is the view definition of the scorecard. There is only one configured view per scorecard.

- **SliceHashCode:** The hash code of the slice within scorecard. The slice hash code is used to retrieve the associated annotation.
- **Slice:** The slice data. It is used in conjunction with column **CellPath** to identify a single cell in a grid.
- **CellPath:** The path to the cell data. It is used in conjunction with column Slice to identify a single cell in a grid.
- **CellPathHashCode:** The hash code of the cell path. The cell path hash code is used to retrieve associated annotation.
- **Created:** The date and time of this annotation's creation.
- **Owner:** The login of the user who created this annotation.
- **LastUpdated:** The date and time of this annotation's latest modification.
- **LastUpdatedBy:** The login of the user who last changed this annotation.
- **SerializedXml:** The XML serialization of an annotation object for a cell's annotation in a scorecard.

3.1.1.2 ParameterValues

The following table describes the abstract data model for **ParameterValues**.

PPSParameterValues		
PK **PK**	**Login** **ParameterUniqueName**	**VARCHAR(2000)** **VARCHAR(4096)**
	LastUpdated **SerializedXml**	**DATETIME** **TEXT(0)**

Note: the column shown in bold style in the diagram indicates the column is required. Otherwise the column is not required.

The protocol server stores all implementation-specific information about selected filters for an element location for a user in the following sets of data. The selected filters are used to pass data from one PerformancePoint first-class object to another in a dashboard and connect a filter to an analytic grid/chart or a scorecard.

ParameterValues Set: A collection of entries corresponding to selected filters for an element location for a user. Each entry MUST be uniquely identified by column **Login** and **ParameterUniqueName**. There is no relationship between this table and other tables. Each user login can contain many entries of selected filters. Each entry MUST include the following elements:

- **Login:** The login name of the user for this user selected filters record.
- **ParameterUniqueName:** The name of the filters, also called filter location, it is a unique identifier for a specific connection between specific PerformancePoint first-class objects.
- **LastUpdated:** The date and time of this user selected filters' latest modification.
- **SerializedXml:** The XML serialization of the single column data table which contains user selected filters for the user login

3.1.1.3 TempFCOs

The following table describes the abstract data model for **TempFCOs**.

PPSTempFCOs		
PK	**ElementId**	**GUID**
	Version	**INTEGER**
	ObjectType	**INTEGER**
	Name	**VARCHAR(1000)**
	Description	VARCHAR(4000)
	Owner	VARBINARY(1000)
	Created	**DATETIME**
	LastAccessed	**DATETIME**
	SerializedXml	**TEXT(0)**

Note: the column shown in bold style in the diagram indicates the column is required. Otherwise the column is not required.

The protocol server stores all implementation-specific information about the temporary copies of PerformancePoint first-class objects in the following sets of data. The table is used when generating analytic grid and chart views in the dashboard designer and when navigating on analytic charts and grids in either the designer or the dashboard.

TempFCOs Set: A collection of entries corresponding to temporary copies of PerformancePoint first-class objects. Each entry MUST be uniquely identified by column **ElementId**. There is no relationship between this table and other tables. Each entry MUST include the following elements:

- **ElementId:** The unique identifier of **TempFCO** record. **TempFCO** is the temporary copies of PerformancePoint first-class objects.
- **Version:** The version of the **TempFCO** record.
- **ObjectType:** The object type of PerformancePoint first-class object for the **TempFCO** record. The only object type which is supported for TempFCO record is temporary report view. There are eight object types of PerformancePoint first-class object: KPI, scorecard, report view, indicator, data source(1), dashboard, temporary report view, and filter.
- **Name:** The name of the **TempFCO** record.
- **Description:** The description of the **TempFCO** record.
- **Owner:** The login of the user who created this **TempFCO** record.
- **Created:** The date and time of this **TempFCO** record creation.
- **LastAccessed:** The name of the user who last read this **TempFCO** record's **SerializedXml** column data.
- **SerializedXml:** The XML serialization of a PerformancePoint first-class object element for **TempFCO** element.

3.1.2 Timers

An execution timeout timer on the protocol server governs the execution time for any requests. The amount of time is specified by a timeout value that is configured on the protocol server for all connections.

3.1.3 Initialization

A connection that uses the underlying protocol layers that are specified in section 1.4 MUST be established before using this protocol.

Listening endpoints are set up on the back end database server to handle inbound PPS application requests.

Authentication of the PPS application connection to the back-end database server MUST occur before this protocol can be used.

The data structures, stored procedures, and actual data are persisted by the back-end database server within databases, so any operations to initialize the state of the database MUST occur before the back-end database server can use this protocol. The data for the PPS server MUST already exist within the back-end database server in a valid state.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

Unless otherwise specified, all stored procedures defined in this section are located in the PerformancePoint Monitoring Service database.

Unless otherwise specified, all stored procedure input parameters MUST NOT be NULL. Because stored procedures use the input parameters for data retrieval from tables, failure to provide valid values will (unless otherwise specified) cause an error as specified in section 3.1.5. Return codes and outputs MUST be handled appropriately by the protocol client or the system behavior is indeterminate.

Unless otherwise specified, all fields returned in the result sets MUST NOT be NULL. If the stored procedures are not provided to the expected calling parameters, or if they do not return the expected result set values, the system behavior is indeterminate.

For the sake of clarity, a name has been assigned to any columns in the result sets that do not have a defined name in their current implementation. This does not affect the operation of the result set, because the ordinal position of any column with no defined name is expected by the protocol client. Such names are designated in the text using curly braces in the form of {*name*}.

This section describes the following stored procedures:

Procedure name	Description
proc_PPS_AddAnnotation	Insert an annotation record of a scorecard's cell's into the **PPSAnnotation** table. If the maximum number of annotations allowed for that scorecard is reached, an error message will be displayed in the server's event log, and the insert operation will be aborted.

Procedure name	Description
proc_PPS_GetAnnotation	Retrieve annotation record(s) based on annotation's unique identifier. This query does not place read lock on the record(s) retrieved.
proc_PPS_AddParameterValue	Insert a user selected filters record into the **PPSParameterValues** table. If an existing **PPSParameterValues** record is found based on the user's login unique identifier and filter's name (that is, **ParameterUniqueName** column value), an update to this matched record will occur. Otherwise, a new record will be inserted into **PPSParameterValues** table.
proc_PPS_AddTempFCO	Insert or update **TempFCO** record. If an existing **TempFCO** record is found based on the **TempFCO** identifier and its **ObjectType** value, an update to this matched record will occur. Otherwise, a new record will be inserted into **PPSTempFCOs** table.
proc_PPS_DropAnnotation	Delete annotation record(s) based on the **Annotation** unique identifier.
proc_PPS_DropParameterValues	Delete all **ParameterValues** records whose **LastUpdated** column values are smaller than a specific expiration date.
proc_PPS_DropTempFCO	Delete one **TempFCO** record based on **TempFCO** unique identifier and **TempFCO**'s **ObjectType** value.
proc_PPS_DropTempFCOs	Delete all **TempFCO** records whose **LastAccessed** column values are smaller than a specific expiration date.
proc_PPS_GetAnnotationBySlice	Retrieve annotation record(s) based on scorecard's unique identifier, cell path hash code, and the slice's hash code. This query does not place read lock on the record(s) retrieved.
proc_PPS_GetAnnotations	Retrieve annotation record(s) based on scorecard's unique identifier. This query does not place read lock on the record(s) retrieved.
proc_PPS_GetParameterValues	Retrieve one **ParameterValues** record based on user's login identifier and a filter's name.
proc_PPS_GetTempFCO	Retrieve one **TempFCO** record based on a **TempFCO** unique identifier and a **TempFCO**'s **ObjectType** value and a **TempFCO**'s version value. The record's **LastAccessed** column is also updated with the date and time when the record is retrieved.
proc_PPS_getTempFCOs	Retrieve all **TempFCO** records based on a **TempFCO**'s **ObjectType** value.
proc_PPS_GetTempFCOVersion	Retrieve one **TempFCO** record's **Version** value based on a **TempFCO** unique identifier and a **TempFCO**'s **ObjectType** value.

Procedure name	Description
proc_PPS_TrimAnnotationsByOwner	Delete annotation record(s) based on an annotation's **Owner** value.
proc_PPS_TrimAnnotationsByScorecardId	Delete annotation record(s) based on a scorecard unique identifier.
proc_PPS_TrimAnnotationsByUntouchedSince	Delete all annotation records whose **LastUpdated** column values are smaller than a specific expiration date.
proc_PPS_UpdateAnnotation	Update annotation record(s) based on an **Annotation**'s unique identifier. It will return an output value called **@LastUpdated** with the current update date and time. If the record is not found, it returns an error message.

3.1.5.1 proc_PPS_AddAnnotation

The **proc_PPS_AddAnnotation** stored procedure is called to insert a scorecard's cell's annotation record into **PPSAnnotation** table (see section 2.2.5.1). If the maximum number of annotations allowed for that scorecard is reached, an error message will be displayed in the server's event log, and the insert operation will be aborted. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_AddAnnotation (
@ScorecardId uniqueidentifier,
@AnnotationId uniqueidentifier,
@ConfiguredViewId uniqueidentifier,
@SliceHashCode int,
@Slice ntext,
@CellPath ntext,
@CellPathHashCode int,
@Owner nvarchar(1000),
@SerializedXml xml,
@ErrorMessage nvarchar(256),
@MaxAnnotationAllowed int,
@Created datetime OUTPUT,

);
```

@ScorecardId: The unique identifier of scorecard. The value MUST NOT be NULL.

@AnnotationId: The unique identifier of the annotation. The value MUST NOT be NULL.

@ConfiguredViewId: The unique identifier of configured view. Configured view is the view definition of the scorecard. There is only one configured view for one scorecard. The value MUST NOT be NULL.

@SliceHashCode: The hash code of the slice within scorecard. The slice hash code is used to retrieve associated annotation. The value MUST NOT be NULL.

@Slice: The slice data. It is used in conjunction with column CellPath to identify a single cell in a grid. The value MUST NOT be NULL.

@CellPath: The cell path data. It is used in conjunction with column Slice to identify a single cell in a grid. The value MUST NOT be NULL.

@CellPathHashCode: The hash code of the cell path. The cell path hash code is used to retrieve associated annotation. The value MUST NOT be NULL.

@Owner: The login of the user who created this annotation. This value MUST NOT be NULL.

@SerializedXml: The serialized XML data which contains a cell's annotation in a scorecard. This value MUST NOT be NULL.

@ErrorMessage: The error message text that will be displayed in the server's event log when the number of annotations for that scorecard has been reached.

@MaxAnnotationAllowed: The maximum number of annotations allowed for that scorecard.

@Created: The date and time of this annotation's creation. This value MUST NOT be NULL.

Output parameter values:

Value	Description
ppsannotations.getdate	The output parameter **@Created** value is set to SQL Server function **getdate** value which is the current date and time value.

Error code values:

Value	Description
50000	If the number of annotations for that scorecard exceeds the limit, this error will show up.

Return Values: An integer which MUST be in the following table.

Value	Description
-1	The insertion of the annotation record for a scorecard's cell failed because the number of annotations allowed for that scorecard was reached.
0	The insertion of the annotation record for a scorecard's cell succeeded.

Result Sets: MUST NOT return any result sets.

3.1.5.2 proc_PPS_GetAnnotation

The **proc_PPS_getAnnotation** stored procedure is called to retrieve annotation record(s) based on annotation's unique identifier. This query does not place read lock(s) on the record(s) retrieved. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_GetAnnotation (
@AnnotationId uniqueidentifier,

);
```

@AnnotationId: The unique identifier of the annotation. The value MUST NOT be NULL.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a proc_PPS_GetAnnotation.ResultSet0

3.1.5.3 proc_PPS_AddParameterValue

The **proc_PPS_AddParameterValue** stored procedure is called to insert or update a user selected filters record in **PPSParameterValues** table (see section 2.2.5.2). If an existing **PPSParameterValues** record is found based on the user's login unique identifier and filter name, that is, **ParameterUniqueName** column value, an update to this matched record will happen. Otherwise, a new record will be inserted into **PPSParameterValues** table. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_AddParameterValue (
@Login nvarchar(1000),
@ParameterUniqueName nvarchar(2048),
@SerializedXml xml,

);
```

@Login: The login name of the user for this user selected filters record. This value MUST NOT be NULL.

@ParameterUniqueName: The name of the filters, also called filter location, it is a unique identifier for a specific connection between specific PerformancePoint first-class objects. This value MUST NOT be NULL.

@SerializedXml: The serialized XML data which contains user selected filters for the login. This value MUST NOT be NULL.

Error code values:

Value	Description
5580001	The update to the existing record in PPSParamaterValues table failed.
5580002	The insertion of a new record to PPSParamaterValues table failed.

Return Values: An integer which MUST be in the following table.

Value	Description
0	The update to the existing record or the insertion of a new record to **PPSParameterValues** table succeeded.
-1	The update to the existing record or the insertion of a new record to **PPSParameterValues** table failed.

Result Sets: MUST NOT return any result sets.

3.1.5.4 proc_PPS_AddTempFCO

The **proc_PPS_AddTempFCO** stored procedure is called to insert a new **TempFCO** record or update an existing **TempFCO** record (see section 2.2.5.3). If an existing **TempFCO** record is found based on the **TempFCO** identifier and its ObjectType value, an update to this matched record will happen. Otherwise, a new record will be inserted into **PPSTempFCOs** table. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_AddTempFCO (
@ElementId uniqueidentifier,
@Version int,
@ObjectType int,
@Name nvarchar(1000),
@Description nvarchar(4000) = null,
@Owner nvarchar(1000) = null,
@SerializedXml xml,

);
```

@ElementId: The unique identifier of **TempFCO** record. **TempFCO** is the temporary copy of PerformancePoint first-class objects. This value MUST NOT be NULL.

@Version: The version of the **TempFCO** record. This value MUST NOT be NULL.

@ObjectType: The object type of PerformancePoint first-class object. The only object type which is supported for **TempFCO** record is temporary report view [MS-PPSAS]. This value MUST NOT be NULL. There are eight object types of PerformancePoint first-class object: **KPI**, scorecard, report view, indicator, data source(1), dashboard, temporary report view, and **filter**.

@Name: The name of the **TempFCO** record. This value MUST NOT be NULL.

@Description: The description of the **TempFCO** record. This value MAY be NULL.

@Owner: The login of the user who created this **TempFCO** record. This value MUST NOT be NULL.

@SerializedXml: The serialized XML data which contains the temporary copy of PerformancePoint first-class object data. The value MUST NOT be NULL. It is a serialized version of PerformancePoint first-class object.

Error code values:

Value	Description
5580001	The update to the existing record in PPSTempFCOs table failed.
5580002	The insertion of a new record to PPSTempFCOs table failed.

Return Values: An integer which MUST be in the following table.

Value	Description
0	The insertion or the update of a TempFCO record into PPSTempFCOs table succeeded.
-1	The insertion or the update of a TempFCO record into PPSTempFCOs table failed.

Result Sets: MUST NOT return any result sets.

3.1.5.5 proc_PPS_DropAnnotation

The **proc_PPS_DropAnnotation** stored procedure is called to delete one annotation record based on an annotation's unique identifier. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_DropAnnotation (
```

```
  @AnnotationId uniqueidentifier,

);
```

@AnnotationId: The unique identifier of the annotation. The value MUST NOT be NULL.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.6 proc_PPS_DropParameterValues

The **proc_PPS_DropParameterValues** stored procedure is called to delete all **ParameterValues** records from **PPSParameterValues** table where **LastUpdated** column values are smaller than the specific expiration date. The T-SQL syntax for the stored procedure is as follows:

```
  PROCEDURE proc_PPS_DropParameterValues (
  @ExpirationDate datetime,

);
```

@ExpirationDate: The expiration date and time which is used to delete ParameterValues records.

Error code values:

Value	Description
5580005	The delete to ParameterValues records in PPSParamaterValues table failed.

Return Values: An integer which MUST be in the following table.

Value	Description
0	The deletion to ParameterValues records in PPSParamaterValues table succeeded.
-1	The deletion to ParameterValues records in PPSParamaterValues table failed.

Result Sets: MUST NOT return any result sets.

3.1.5.7 proc_PPS_DropTempFCO

The **proc_PPS_DropTempFCO** stored procedure is called to delete one **TempFCO** record based on a **TempFCO**'s unique identifier and a **TempFCO**'s **ObjectType** value. The T-SQL syntax for the stored procedure is as follows:

```
  PROCEDURE proc_PPS_DropTempFCO (
  @ElementId uniqueidentifier,
  @ObjectType int,

);
```

@ElementId: The unique identifier of a TempFCO record. TempFCO is the temporary copy of PerformancePoint first-class object. This value MUST NOT be NULL.

@ObjectType: The object type of PerformancePoint first-class object. The only object type that is supported for **TempFCO** record is temporary report view, as specified in [MS-PPSAS]. This value MUST NOT be NULL. There are eight object types of PerformancePoint first-class object: KPI, scorecard, report view, indicator, data source(1), dashboard, temporary report view, and filter.

Error code values:

Value	Description
5580005	The deletion of TempFCO record from PPSTempFCOs table failed.

Return Values: An integer which MUST be in the following table.

Value	Description
0	The deletion of TempFCO record from PPSTempFCOs table succeeded.
-1	The deletion of TempFCO record from PPSTempFCOs table failed.

Result Sets: MUST NOT return any result sets.

3.1.5.8 proc_PPS_DropTempFCOs

The **proc_PPS_DropTempFCOs** stored procedure is called to delete all **TempFCO** records where **LastAccessed** column values are smaller than a specific expiration date. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_DropTempFCOs (
@ExpirationDate datetime,

);
```

@ExpirationDate: The expiration date and time, which is used to delete **TempFCO** records from **PPSTempFCOs** table.

Error code values:

Value	Description
5580005	The deletion of **TempFCO** record(s) from **PPSTempFCOs** table failed.

Return Values: An integer which MUST be in the following table.

Value	Description
0	The deletion of **TempFCO** record(s) from **PPSTempFCOs** table succeeded.
-1	The deletion of **TempFCO** record(s) from **PPSTempFCOs** table failed.

Result Sets: MUST NOT return any result sets.

3.1.5.9 proc_PPS_GetAnnotationBySlice

The **proc_PPS_GetAnnotationBySlice** stored procedure is called to retrieve annotation record(s) based on the scorecard's unique identifier, cell path hash code, and the slice's hash code. This query does not place read lock on the record(s) retrieved. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_GetAnnotationBySlice (
@ScorecardId uniqueidentifier,
@CellPathHashCode int,
@SliceHashCode int,

);
```

@ScorecardId: The unique identifier of the scorecard. The value MUST NOT be NULL.

@CellPathHashCode: The hash code of the cell path. The cell path hash code is used to retrieve associated annotation. The value MUST NOT be NULL.

@SliceHashCode: The hash code of the slice within the scorecard. The slice hash code is used to retrieve associated annotation. The value MUST NOT be NULL.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a proc_PPS_GetAnnotationBySlice.ResultSet0

3.1.5.10 proc_PPS_GetAnnotations

The **proc_PPS_GetAnnotations** stored procedure is called to retrieve annotation record(s) based on the scorecard's unique identifier. This query does not place read lock(s) on the record(s) retrieved. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_GetAnnotations (
@ScorecardId uniqueidentifier,

);
```

@ScorecardId: The unique identifier of the scorecard. The value MUST NOT be NULL.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a proc_PPS_GetAnnotations.ResultSet0

3.1.5.11 proc_PPS_GetParameterValue

The **proc_PPS_GetParameterValues** stored procedure is called to retrieve one **ParameterValues** record from **PPSParameterValues** table based on user's login identifier and filter's name. **ParameterValues** record contains user selected filters for the login. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_GetParameterValue (
```

```
@Login nvarchar(1000),
@ParameterUniqueName nvarchar(2048),
```

);

@Login: The login name of the user for this user selected filters record. This value MUST NOT be NULL.

@ParameterUniqueName: The name of the filters. Also called filter location. It is a unique identifier for a specific connection between specific PerformancePoint first-class object. This value MUST NOT be NULL.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a proc_PPS_GetParameterValue.ResultSet0

3.1.5.12 proc_PPS_GetTempFCO

The **proc_PPS_GetTempFCO** stored procedure is called to retrieve one **TempFCO** record based on **TempFCO** unique identifier and **TempFCO**'s **ObjectType** value and **TempFCO**'s version value. The record's **LastAccessed** column is also updated with the date and time when the record is retrieved. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_GetTempFCO (
@ElementId uniqueidentifier,
@Version int,
@ObjectType int,
```

);

@ElementId: The unique identifier of **TempFCO** record. **TempFCO** is the temporary copies of PerformancePoint first-class objects. This value MUST NOT be NULL.

@Version: The version of the **TempFCO** record. This value MUST NOT be NULL.

@ObjectType: The object type of PerformancePoint first-class object. The only object type which is supported for **TempFCO** record is temporary report view. This value MUST NOT be NULL. There are eight object types of PerformancePoint first-class object: KPI, scorecard, report view, indicator, data source(1), dashboard, temporary report view, and filter.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a proc_PPS_GetTempFCO.ResultSet0

3.1.5.13 proc_PPS_GetTempFCOs

The **proc_PPS_getTempFCOs** stored procedure is called to retrieve all **TempFCO** records from **PPSTempFCOs** table based on **TempFCO**'s **ObjectType** value. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_GetTempFCOs (
```

```
@ObjectType int,

);
```

@ObjectType: The object type of PerformancePoint first-class object. The only object type which is supported for **TempFCO** record is temporary report view. This value MUST NOT be NULL. There are eight object types of PerformancePoint first-class object: KPI, scorecard, report view, indicator, data source(1), dashboard, temporary report view, and filter.

Return Values: An integer which MUST be 0.

Result Sets:

This stored procedure MUST return a proc_PPS_GetTempFCOs.ResultSet0

3.1.5.14 proc_PPS_GetTempFCOVersion

The **proc_PPS_getTempFCOVersion** stored procedure is called to retrieve one **TempFCO** record's **Version** value based on **TempFCO** unique identifier and **TempFCO**'s **ObjectType** value. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_GetTempFCOVersion (
@ElementId uniqueidentifier,
@ObjectType int,
@Version int OUTPUT,

);
```

@ElementId: The unique identifier of **TempFCO** record. **TempFCO** is the temporary copies of PerformancePoint first-class object. This value MUST NOT be NULL.

@ObjectType: The object type of PerformancePoint first-class object. The only object type which is supported for **TempFCO** record is temporary report view. This value MUST NOT be NULL. There are eight object types of PerformancePoint first-class object: KPI, scorecard, report view, indicator, data source(1), dashboard, temporary report view, and filter.

@Version: The version of the **TempFCO** record. This value MUST NOT be NULL. It is an output parameter.

Output parameter values:

Value	Description
ppstempfcos.version	The output parameter **@Version** is set to the **Version** value retrieved from **TempFCO** record.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.15 proc_PPS_TrimAnnotationsByOwner

The **proc_PPS_TrimAnnotationsByOwner** stored procedure is called to delete annotation record(s) based on annotation's **Owner** value. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_TrimAnnotationsByOwner (
@Owner nvarchar(1000),

);
```

@Owner: The login of the user who created this annotation. This value MUST NOT be NULL.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.16 proc_PPS_TrimAnnotationsByScorecardId

The **proc_PPS_TrimAnnotationsByScorecardId** stored procedure is called to delete annotation record(s) based on scorecard unique identifier. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_TrimAnnotationsByScorecardId (
@ScorecardId uniqueidentifier,

);
```

@ScorecardId: The unique identifier of the scorecard. The value MUST NOT be NULL.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.17 proc_PPS_TrimAnnotationsByUntouchedSince

The **proc_PPS_TrimAnnotationsByUntouchedSince** stored procedure is called to delete all annotation records where **LastUpdated** column values are smaller than the specific expiration date. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_TrimAnnotationsByUntouchedSince (
@UntouchedSince datetime,

);
```

@UntouchedSince: The expiration date and time which is used to delete annotation records.

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.18 proc_PPS_UpdateAnnotation

The **proc_PPS_UpdateAnnotation** stored procedure is called to update annotation record(s) based on annotation's unique identifier. It will return an output value called **@LastUpdated** with the current update date and time. If the record is not found, it returns an error message. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_PPS_UpdateAnnotation (
@AnnotationId uniqueidentifier,
@Owner nvarchar(1000),
@LastUpdatedBy nvarchar(1000),
@LastUpdated datetime OUTPUT,
@SerializedXml xml,
@ErrorMessage nvarchar(256),

);
```

@AnnotationId: The unique identifier of the annotation. The value MUST NOT be NULL.

@Owner: The login of the user who created this annotation. This value MUST NOT be NULL.

@LastUpdatedBy: The login of the user who last changed this annotation. This value MUST NOT be NULL.

@LastUpdated: The date and time of this annotation's latest modification. This value MUST NOT be NULL.

Output parameter values:

Value	Description
getdate	The output parameter @LastUpdated value is set to SQL Server function getdate() value which is the current date and time value.

@SerializedXml: The serialized XML data which contains a cell's annotation in a scorecard. This value MUST NOT be NULL.

@ErrorMessage: The error message text that will be displayed in the server's event log when the record for update is not found.

Error code values:

Value	Description
50000	If record for update is not found, this message will show up.

Return Values: An integer which MUST be in the following table.

Value	Description
-1	The record for update is not found. Update annotation operation failed.
0	The update of an annotation record based on the annotation identifier succeeded.

Result Sets: MUST NOT return any result sets.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Client Details

None.

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

None.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

This section provides specific example scenarios for end-to-end administration tasks.

Security for this protocol is controlled by the permissions to the databases on the back-end database server, which is negotiated as part of the PerformancePoint Services ApplicationServer Protocol, as described in [MS-PPSAPP].

This protocol specifies the communication requests between the front-end Web server and the back-end database server to perform these tasks: maintain scorecard annotations, pass user selected filters between PerformancePoint first-class objects in a dashboard and connect a filter to an analytic grid/chart or a scorecard, and maintain state information when navigating among PerformancePoint first-class objects.

This protocol enables a protocol client to do the following:

- Add, change, retrieve and delete scorecard annotations from the database on the back-end database server**.**
- Add, retrieve, and delete user selected filters for a user from the database on the back-end database server**.**
- Add, retrieve, and delete temporary copies of PerformancePoint first-class object from the database on the back-end database server**.**

Some protocol examples are documented in the following sections.

4.1 Scorecard Annotation Administration

This section illustrates the protocol operations required to add, change, retrieve and delete scorecard annotations from the database on the back-end database server. The following figure shows all the available operations in regards to scorecard annotation handling.

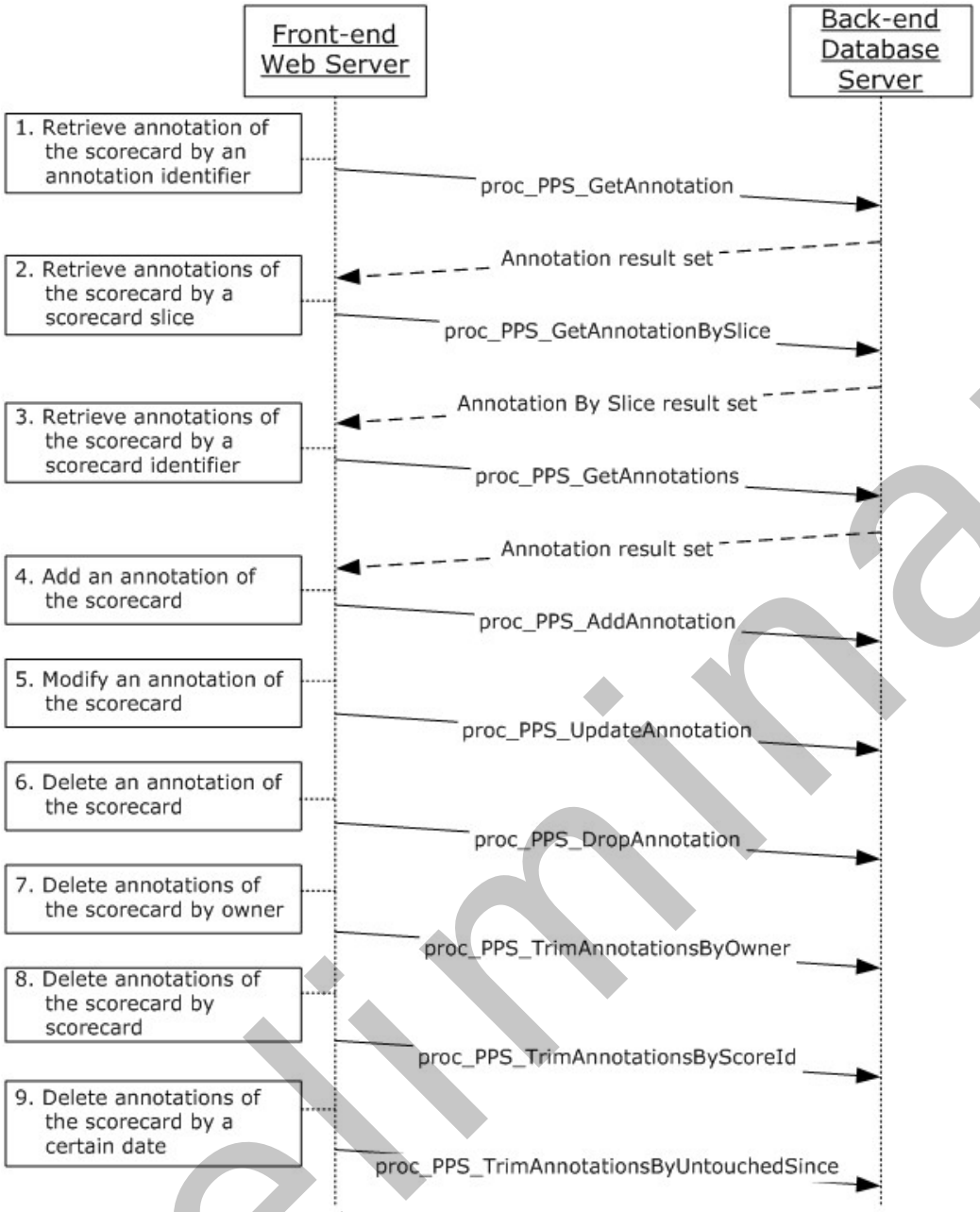

Figure 5: Scorecard annotations administration

The available operations in the preceding diagram are explained as follows:

1. The protocol client retrieving an annotation of the scorecard by the annotation identifier.
2. The protocol client retrieves annotations of the scorecard by a scorecard slice.
3. The protocol client retrieves annotations of the scorecard by a scorecard identifier.

4. The protocol client calls **proc_PPS_AddAnnotation** stored procedure to add a new annotation of scorecard into PPSAnnotations table.

5. To change data for the existing annotation of the scorecard**,** the protocol client calls **proc_PPS_UpdateAnnotation** stored procedure.

6. The client deletes an annotation of the scorecard.

7. The client deletes annotations of the scorecard by the owner identifier.

8. The client deletes annotations of the scorecard by the scorecard identifier.

9. The client deletes annotations of the scorecard which have not been accessed since a certain date.

The following user scenarios illustrate the usage of some protocol operations shown earlier.

Example 1: User views a scorecard from the **site (2)**, and then selects a specific cell and enters two annotations to this cell.

The following is what happened on the system:

The client side on the front-end web server sends request to the back end database server to add annotations for the scorecard cell. Server side calls stored procedure **proc_PPS_AddAnnotation** to add the annotations for that cell in the scorecard into database server table.

Example 2: The administrator goes to the Central Administration site to delete scorecard annotations that have not been updated since a specific date.

The following is what happened on the system:

The client side on the front-end web server sends request to the back end database server to delete scorecard annotations which have not been updated since a specific date. Server side calls the **proc_PPS_TrimAnnotationsByUntouchedSince** stored procedure to perform this task from the backend database server.

4.2 User-Selected Filter Administration

One typical use of this protocol is a service application that is called by one or more other service applications, is to pass user selected filters between PerformancePoint first-class objects in a dashboard and connect a filter to an analytic grid/chart or a scorecard.

This section illustrates the protocol operations required to add, retrieve, and delete user selected filters for a user from the database on the back-end database server**.** The following figure shows all the available operations in regards to the administration of user selected filters for a user.

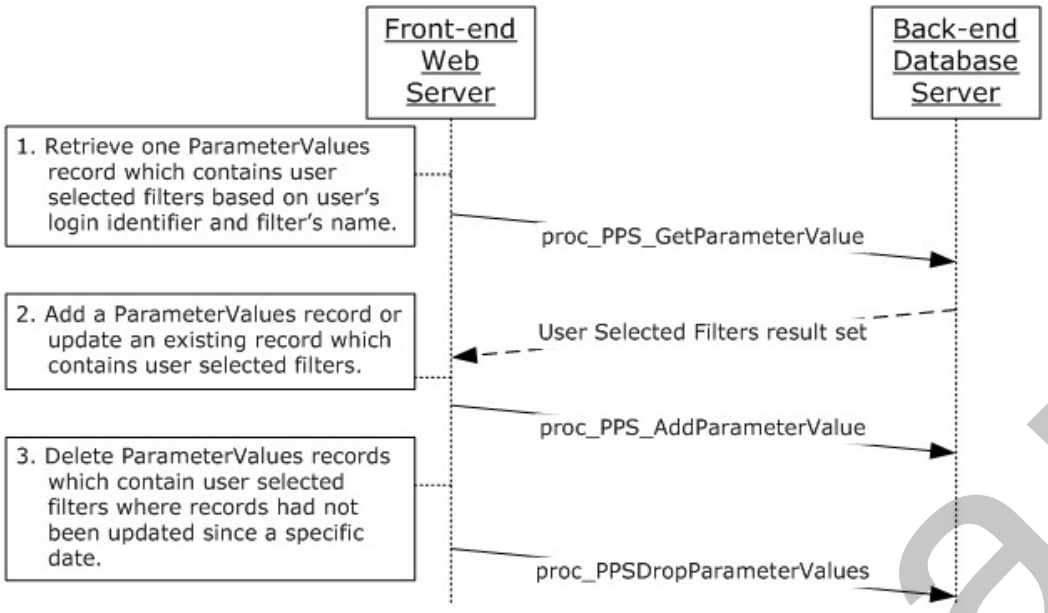

Figure 6: User Selected Filters Administration

The available operations in the preceding diagram are explained as follows:

1. The protocol client retrieves one ParameterValues record which contains user selected filters based on a user's login identifier and a filter's name.

2. The protocol client calls **proc_PPS_AddParameterValue** stored procedure to add a new ParameterValues record, which contains user selected filters, into the PPSParameterValues table. If the record exists, the stored procedure will update to the existing record.

3. The client deletes ParameterValues records, which contain user selected filters for records that had not been updated since a specific date.

The following user scenarios illustrate the usage of some protocol operations shown earlier.

Example 1: The user creates a dashboard with a filter, and then makes some choices with that filter.

The following is what happened at the system:

The client side on the front-end web server sends request to the back end database server to save the user selected filters to the database table. The database server side calls the **proc_PPS_AddParameterValue** stored procedure to add user selected filters into database server table.

Example 2: The user opens a new browser and views the dashboard created in Example 1. The selection that the user made in Example 1 is displayed here.

The following is what happened at the system:

The client side on the front-end web server sends request to the back end database server to query the user selected filters from the database table. The database server side calls the

proc_PPS_GetParameterValue stored procedure to return the user selected filters from database server table.

4.3 Temporary Copies of PerformancePoint First-Class Object Administration

A typical use of this protocol is a service application that is called by one or more other service applications, is to maintain state information when navigating among PerformancePoint first-class objects for a certain period of time, and deleting the state information after the time expires.

This section illustrates the protocol operations required to add, retrieve, and delete temporary copies of PerformancePoint first-class object from the database on the back-end database server**.** The following diagram shows all the available operations in regards to the administration of temporary copies of PerformancePoint first-class object.

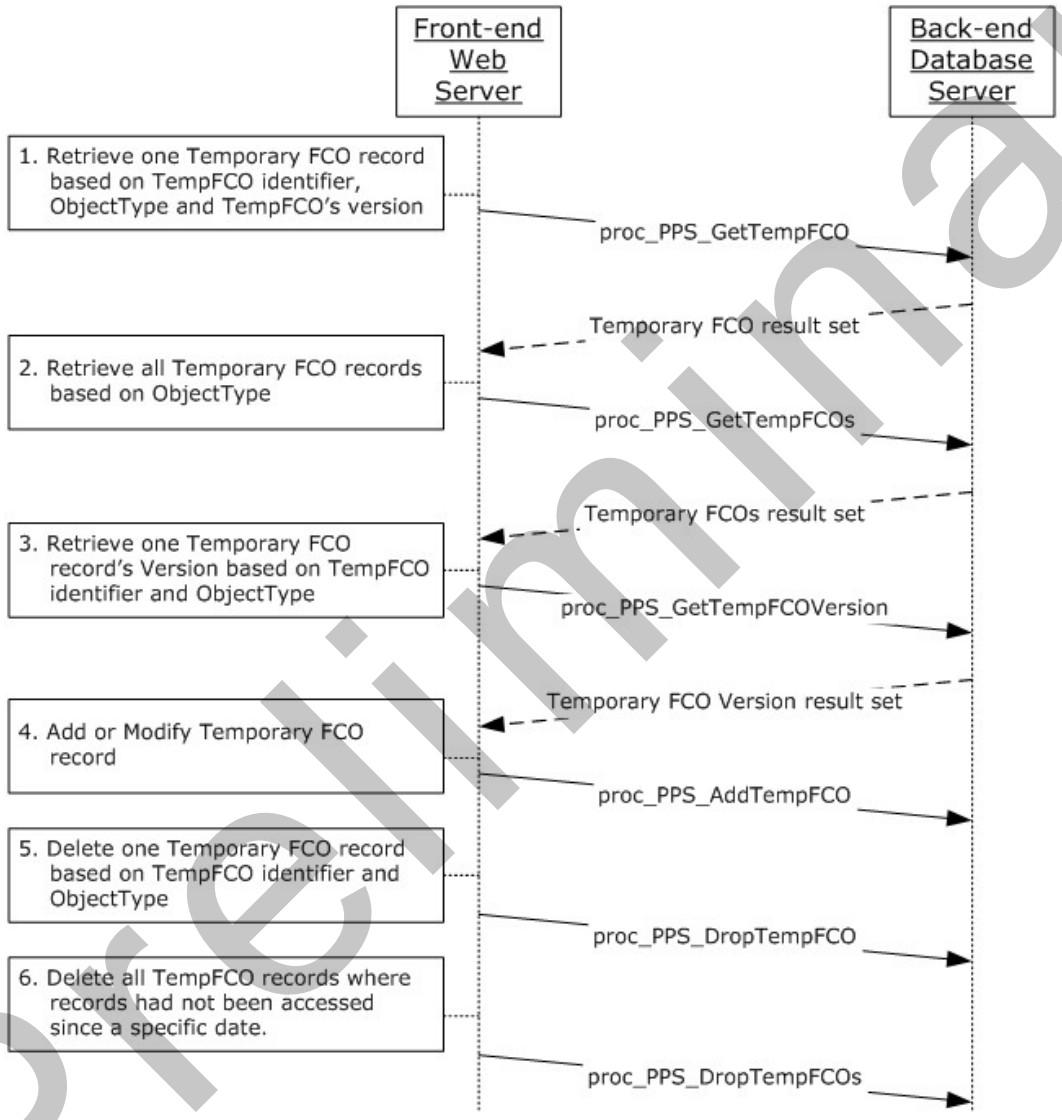

Figure 7: Temporary FCOs administration

The available operations in the preceding diagram are explained as follows:

1. The protocol client retrieves one **TempFCO** record based on a **TempFCO** unique identifier and a TempFCO's version.

2. The protocol client retrieves all **TempFCO** records based on a **TempFCO**'s **ObjectType**.

3. The protocol client retrieves one **TempFCO** record's version based on the **TempFCO** unique identifier and the **TempFCO**'s **ObjectType**.

4. The protocol client calls **proc_PPS_AddTempFCO** stored procedure to add a new **TempFCO** record into **PPSTempFCOs** table. If the record exists, the stored procedure will update to the existing record.

5. The client deletes one **TempFCO** record based on the **TempFCO** unique identifier and the **TempFCO**'s **ObjectType**.

6. The client deletes all **TempFCO** records that had not been accessed since a specific date.

Example 1: The user views a published chart from Dashboard Designer, and then drills down on an item, for example, a bar, in the chart.

The following is what happened at the system:

The client side on the front-end web server sends request to the back end database server to save the state information of the PerformancePoint first-class object to the database table. The database server side calls the **proc_PPS_AddTempFCO** stored procedure to add such state information into database server table. After certain period of time (this is configurable parameter), such state information will be deleted from the backend database server.

5 Security

5.1 Security Considerations for Implementers

The permissions to data accessed through these procedures needs to be validated prior to retrieving any content.

The security of a temporary copy of PerformancePoint first-class object is based on the parent underlying protocol and defined by the ElementId or ScorecardId.

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® Server 2010
- Microsoft® SharePoint® Server 2013 Preview

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.

7 Change Tracking

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

8 Index

N

O

P

R

S

T

V

X