SW Design	BOSCH
Title of document:	Interface Specification For Receiving Events from PV System (WCF)
Type:	API document
Release of document:	
Date:	7th Oct 2022
Project:	PowerVision
Component:	Event Notification
SW Version:	1.0.0.11

SW design description	
State:	[x] In Progress [x] Discussed [] Accepted
Author:	Praveenkumar Bouna
Review:	Narendra Kumar
Version history: 1.0.0.3 – 21st Nov 2019 – Support for WCF 1.0.0.11 – 7th Oct 2022 – Support for transaction restriction.	

Contents

1 Introduction

PowerVision (PV) system raises events to 3rd party systems whenever the configured alarm/event triggers within the PV system. The 3rd party system must have their web service with below mentioned API which accepts the mentioned list of parameters. APIs are exposed through WCF NetTcp contracts. The usage of NetTcp binding assists in maintaining single session between the client and server, and also results in faster processing of sending alarms from the server to the clients.

Note that if the session gets disconnected due to network issue or any other reasons, then client needs to reconnect to the server. The client can use Closed or Faulted events to track such situations.

2 WCF Contract

PowerVision system will act as the Server and 3rd party system acts as a Client. The event will be raised from PowerVision system to the 3rd party application through WCF call back.

The list of events to be notified will be configurable in the PV system.

2.1 Authentication

The system uses WCF NetTcp Binding to host its service. UserNamePasswordValidator will be used at the service to authenticate the client request. The client needs to set Username and Password during the session creation. We need to add this user account in the PV system.

The service is hosted under the **port 11127**.

2.2 IPVEventContract

This interface defines the duplex contract which the client can access to communicate with the server. This service contract will be **implemented at the Server end**.

The function “Initialize” is used to identify the client application within the system. The Client has to first call this function with their application name during initialization.

The function “IsAlive” returns “true” which can be used by the client to check the session with the server.

```
[ServiceContract(Name = "IPVEventContract", CallbackContract = typeof(IPVEventCallback))]
public interface IPVEventContract : IChannelContract
{
    [OperationContract(IsOneWay = true)]
    void Initialize(string clientApplicationName);

    [OperationContract(IsOneWay = false)]
    bool IsAlive();
}

public interface IChannelContract
{
}
```

2.3 IPVEventCallback

Whenever an event is triggered within the PV system, it fires RaiseEvent() through WCF callback to the clients. The **client system has to implement this operation** to get the notification.

```
public interface IPVEventCallback : ICallbackContract
{
    [OperationContract(IsOneWay = true)]
    void RaiseEvent(PVEvent eventDetails);
}

public interface ICallbackContract
{
}
```

2.4 PVEvent

The RaiseEvent callback function sends the event details through PVEvent object parameter. The following are the properties along with the expected values.

```
public class PVEvent
{
    [DataMember]
    public string Name { get; set; }  // Card name. Ignore. Will be blank.
    [DataMember]
    public string Company { get; set; }  // Ignore. Will be blank.
    [DataMember]
    public string UnitNo{ get; set; } // Ignore. Will be blank.
    [DataMember]
    public string Site{ get; set; } // Site name. Will be blank.
    [DataMember]
    public string MonitorPointId { get; set; } // Reader ID. Will be blank.
    [DataMember]
    public string MonitorPointDesc { get; set; } // Reader name
    [DataMember]
    public string EventDesc { get; set; } // Event description/name
    [DataMember]
    public string CardId { get; set; } // Card number
    [DataMember]
    public string Id { get; set; } // Ignore. Will be blank.
    [DataMember]
    public string EventDate { get; set; } // Event occurred date
    [DataMember]
    public string EventTime { get; set; } // Event occurred time
    [DataMember]
    public int AlarmStatus { get; set; } // Ignore. Will be blank.
                                        Alarm status:Normal = 0, Alarm = 3 or 2 or 1.
                                        3 = When Alarm triggered,
                                        2 = Alarm Acknowledged but not cleared,
                                        1 = Alarm cleared but not acknowledged
    [DataMember]
    public int DoorStatus { get; set; } // Ignore. Will be blank.
                                        Door open status (Open = 1, Close = 2)}
```

3 Parameter Formats

Following are the formats of some of the properties:

Parameter	**Format**
EventDate	yyyy-MM-dd
EventTime	HH:mm:ss
MonitorPointId	sss-ttt-ccc-aaa where sss = Site ID ttt = TCU number, ccc = CRC number, aaa = ACU number