Synchronization

Connecting to the SRP Synchronization Server

One of the key benefits of the SRP Schedule Control is that ability to synchronization instances of the control on remote machines. When a user moves an appointment on one machine, all other uses should see that appointment move as well. This is accomplished by connected your SRP Schedule Control to the SRP Synchronization Server.

Connecting With the SRP Schedule Control

Once you've displayed a form with your control, synchronizing controls is easy. Simply set the Server property, and presuming you've passed the correct IP address and port number, the control will handle the rest. All you have to do is set your control properties as usual and the control will send data to the server to be synchronized with other controls.

Connecting Without the SRP Schedule Control

So what do you do if you have a data entry window sans the SRP Schedule Control that contains data you want synchronized to everyone else? The answer is COM. When you register the SRPControls.ocx or the SRPSchedule.ocx, a special COM object is made available to you, which you can use to send data to the SRP Synchronization Server directly. **NOTE:** the COM object is only available on SRP Schedule Control 3.0 or later.

The ProgID of this COM object is "SRP.ScheduleSync". Before you call one of the methods listed here, you first must create an instance of the COM object itself:

```
ServerSendObj = OleCreateInstance("SRP.ScheduleSync")
```

Once you've successfully created the object, then you can use the OleCallMethod routine to call the methods. For example, this is how you would call the Validate method:

```
OleCallMethod(ServerSendObj, "Validate", "127.0.0.1", "25000")
```

Here are the methods you can call:

LockAppt

```
OleCallMethod(Obj, "LockAppt", User, Server, Port, ApptID)
```

The LockAppt method locks an appointment across all synchronized SRP Schedule Controls. This has the same effect as the SRP Schedule Control's Appt Locked property. Like all SRPSchedule methods, you should call this only when the SRP Schedule Control is unavailable.

The User, Server, and Port parameters provide the information needed to find the SRP Synchronization Server. The User parameter can be any string that identifies the user or machine. The Server is the IP or URL address describing the server's location on the network. The Port is the port upon which the SRP Synchronization Server is listening.

The ApptID is the unique ID of the appointment to be locked. If the appointment doesn't exist, then nothing happens. If the appointment does exist, then the SRP Synchronization Server will lock the appointment on all connected SRP Schedule Controls. When this happens, all users will see the appointment with a padlock icon. It will not be unlocked until UnlockAppt is called or the ApptLocked property on any connected SRP Schedule Control is set to 0.

```
// Lock an appt
If Get_Property(OleScheduleCtrl$, "OLE.ProgID") NE "" then
   Set_Property(OleScheduleCtrl$, "OLE.ApptLocked[":ApptID:"]", 1)
end else
   ServerSendObj = OleCreateInstance("SRP.ScheduleSync")
   If OleStatus() else
       OleCallMethod(ServerSendObj, "LockAppt", "Bob", "127.0.0.1", "25000", ApptID)
   end
end
```

UnlockAppt

```
OleCallMethod(Obj, "UnlockAppt", User, Server, Port, ApptID)
```

The LockAppt method unlocks an appointment across all synchronized SRP Schedule Controls. This has the same effect as setting the SRP Schedule Control's ApptLocked property to 0. Like all SRPSchedule methods, you should call this only when the SRP Schedule Control is unavailable.

The User, Server, and Port parameters provide the information needed to find the SRP Synchronization Server. The User parameter can be any string that identifies the user or machine. The Server is the IP or URL address describing the server's location on the network. The Port is the port upon which the SRP Synchronization Server is listening.

The ApptID is the unique ID of the appointment to be unlocked. If the appointment doesn't exist, then nothing happens. If the appointment does exist, then the SRP Synchronization Server will unlock the appointment on all connected SRP Schedule Controls. When this happens, the padlock icon is removed and the users can resize and move the appointment. An appointment is locked when calling either LockAppt or setting the SRP Schedule Control's ApptLocked property to 1.

```
// Unlock an appt
If Get_Property(OleScheduleCtrl$, "OLE.ProgID") NE "" then
   Set_Property(OleScheduleCtrl$, "OLE.ApptLocked[":ApptID:"]", 0)
end else
   ServerSendObj = OleCreateInstance("SRP.ScheduleSync")
   If OleStatus() else
       OleCallMethod(ServerSendObj, "UnlockAppt", "Bob", "127.0.0.1", "25000", ApptID)
   end
end
```

NewAppt

```
OleCallMethod(Obj, "NewAppt", User, Server, Port, ApptInfo)
```

The NewAppt method adds a single new appointment to all synchronized SRP Schedule Controls. This has the same effect as the SRP Schedule Control's AddAppts method. Like all SRPSchedule methods, you should call this only when the SRP Schedule Control is unavailable. This is useful for updating remote schedules when a user creates a new appointment via an OpenInsight form.

The User, Server, and Port parameters provide the information needed to find the SRP Synchronization Server. The User parameter can be any string that identifies the user or machine. The Server is the IP or URL address describing the server's location on the network. The Port is the port upon which the SRP Synchronization Server is listening.

The ApptInfo is a multi-valued parameter describing the new appointment. See the AppointmentList property for details on this structure.

If the appointment already exists, then nothing happens. If the appointment does not exist, then the SRP Synchronization Server will add the appointment to all connected SRP Schedule Controls. You can only add one appointment at a time, unlike the SRP Schedule Control's AddAppts method, which lets you add multiple appointments. For this reason, a single appointment is @FM delimited, whereas AddAppts requires that appointments be @VM delimited.

```
// Add an appt: ApptInfo is @FM delimited
If Get_Property(OleScheduleCtrl$, "OLE.ProgID") NE "" then
   Convert @FM to @VM in ApptInfo
   Send_Message(OleScheduleCtrl$, "OLE.AddAppts", ApptInfo)
end else
   ServerSendObj = OleCreateInstance("SRP.ScheduleSync")
   If OleStatus() else
       OleCallMethod(ServerSendObj, "NewAppt", "Bob", "127.0.0.1", "25000", ApptInfo)
   end
end
```

RemoveAppt

```
OleCallMethod(Obj, "RemoveAppt", User, Server, Port, ApptID)
```

The RemoveAppt method removes a single appointment from all synchronized SRP Schedule Controls. This has the same effect as the SRP Schedule Control's RemoveAppts method. Like all SRPSchedule methods, you should call this only when the SRP Schedule Control is unavailable. This is useful for updating remote schedules when a user deletes an appointment via an OpenInsight form.

The User, Server, and Port parameters provide the information needed to find the SRP Synchronization Server. The User parameter can be any string that identifies the user or machine. The Server is the IP or URL address describing the server's location on the network. The Port is the port upon which the SRP Synchronization Server is listening.

The ApptID is the unique ID of the appointment to be removed. If the appointment doesn't exist, then nothing happens. If the appointment does exist, then the SRP Synchronization Server will remove the appointment on all connected SRP Schedule Controls. You can only remove one appointment at a time, unlike the SRP Schedule Control's RemoveAppts method, which lets you remove multiple appointments.

```
// Remove an appt
If Get_Property(OleScheduleCtrl$, "OLE.ProgID") NE "" then
   Send_Message(OleScheduleCtrl$, "OLE.RemoveAppts", ApptID)
end else
   ServerSendObj = OleCreateInstance("SRP.ScheduleSync")
   If OleStatus() else
       OleCallMethod(ServerSendObj, "RemoveAppt", "Bob", "127.0.0.1", "25000", ApptID)
   end
end
```

UpdateAppt

```
OleCallMethod(Obj, "UpdateAppt", User, Server, Port, ApptInfo)
```

The UpdateAppt method updates a single existing appointment for all synchronized SRP Schedule Controls. This has the same effect as the SRP Schedule Control's Appt property. Like all SRPSchedule methods, you should call this only when the SRP Schedule Control is unavailable. This is useful for updating remote schedules when a user modifies an appointment via an OpenInsight form.

The User, Server, and Port parameters provide the information needed to find the SRP Synchronization Server. The User parameter can be any string that identifies the user or machine. The Server is the IP or URL address describing the server's location on the network. The Port is the port upon which the SRP Synchronization Server is listening.

The ApptInfo is a multi-valued parameter describing the appointment. See the AppointmentList property for details on this structure.

If the appointment doesn't exist, then nothing happens. If the appointment does exist, then the SRP Synchronization Server will update the appointment for all connected SRP Schedule Controls. You can only update one appointment at a time.

```
// Update an appt: ApptInfo is @FM delimited
If Get_Property(OleScheduleCtrl$, "OLE.ProgID") NE "" then
   ApptID = ApptInfo<2>
   ApptInfo = Delete(ApptInfo, 2, 0, 0)
   Set_Property(OleScheduleCtrl$, "OLE.Appt[":ApptID:"]", ApptInfo)
end else
   ServerSendObj = OleCreateInstance("SRP.ScheduleSync")
   If OleStatus() else
       OleCallMethod(ServerSendObj, "UpdateAppt", "Bob", "127.0.0.1", "25000", ApptInfo)
   end
end
```

UpdateEntity

```
OleCallMethod(Obj, "UpdateEntity", User, Server, Port, EntityInfo)
```

The UpdateEntity method updates a single existing entity for all synchronized SRP Schedule Controls. This has the same effect as the SRP Schedule Control's Entity property. Like all SRPSchedule methods, you should call this only when the SRP Schedule Control is unavailable. This is useful for updating remote schedules when a user modifies an appointment via an OpenInsight form.

The User, Server, and Port parameters provide the information needed to find the SRP Synchronization Server. The User parameter can be any string that identifies the user or machine. The Server is the IP or URL address describing the server's location on the network. The Port is the port upon which the SRP Synchronization Server is listening.

The EntityInfo is a multi-valued parameter describing the entity. See the EntityList property for details on this structure.

If the entity doesn't exist, then nothing happens. If the entity does exist, then the SRP Synchronization Server will update the entity for all connected SRP Schedule Controls. You can only update one entity at a time.

```
// Update an entity: EntityInfo is @FM delimited
If Get_Property(OleScheduleCtrl$, "OLE.ProgID") NE "" then
   EntityID = EntityInfo<1>
   EntityInfo = Delete(EntityInfo, 1, 0, 0)
   Set_Property(OleScheduleCtrl$, "OLE.Entity[":EntityID:"]", EntityInfo)
end else
   ServerSendObj = OleCreateInstance("SRP.ScheduleSync")
   If OleStatus() else
       OleCallMethod(ServerSendObj, "UpdateEntity", "Bob", "127.0.0.1", "25000", EntityInfo)
   end
end
```

UpdateApptFlag

```
OleCallMethod(Obj, "UpdateApptFlag", User, Server, Port, FlagInfo)
```

The UpdateApptFlag method updates a single appointment's flag for all synchronized SRP Schedule Controls. This has the same effect as the SRP Schedule Control's ApptFlag property. Like all SRPSchedule methods, you should call this only when the SRP Schedule Control is unavailable. This is useful for updating remote schedules when a user modifies an appointment via an OpenInsight form.

The User, Server, and Port parameters provide the information needed to find the SRP Synchronization Server. The User parameter can be any string that identifies the user or machine. The Server is the IP or URL address describing the server's location on the network. The Port is the port upon which the SRP Synchronization Server is listening.

The FlagInfo is a multi-valued parameter describing the appointment flag to be updated. It is composed of three fields:

Pos	Name	Type	Description
<1>	Appt ID	Text	The appointment whose flag to change
<2>	Flag Index	Integer	The flag to set, 1 - 16
<3>	Flag Value	Boolean	The flag's new value

If the appointment doesn't exist, then nothing happens. If the appointment does exist, then the SRP Synchronization Server will update the appointment flag for all connected SRP Schedule Controls. You can only update one appointment flag at a time.

```
// Update an appt flag, setting flag 4 on
If Get_Property(OleScheduleCtrl$, "OLE.ProgID") NE "" then
   Set_Property(OleScheduleCtrl$, "OLE.ApptFlag[":ApptID:";":4:"]", 1)
end else
   ServerSendObj = OleCreateInstance("SRP.ScheduleSync")
   If OleStatus() else
       FlagInfo = ApptID:@FM:4:@FM:1
       OleCallMethod(ServerSendObj, "UpdateApptFlag", "Bob", "127.0.0.1", "25000", FlagInfo)
   end
end
```

Validate

```
Result = OleCallMethod(Obj, "Validate", Server, Port)
```

The Validate method verifies that the SRP Synchronization Server is running. It does this by connecting to the server, sending a validation request, verifying a response, and disconnecting. If any of these steps fail, then the the result is 0. If all steps succeed, then the result is 1.

The Server, and Port parameters provide the information needed to find the SRP Synchronization Server. The Server is the IP or URL address describing the server's location on the network. The Port is the port upon which the SRP Synchronization Server is listening.

NOTE: Avoid calling the Validate method frequently since doing so takes several seconds per call (if the server is not running). It's always best to check if the server is running early in the application and cache the results.

```
// Validate the server
If Get_Property(OleScheduleCtrl$, "OLE.ProgID") NE "" then
   IsValid = Get_Property(OleScheduleCtrl$, "OLE.ServerValid"
end else
   ServerSendObj = OleCreateInstance("SRP.ScheduleSync")
   If OleStatus() else
       OleCallMethod(ServerSendObj, "Validate", "127.0.0.1", "25000")
   end
end
```

Quit

```
OleCallMethod(Obj, "Quit", Server, Port)
```

The Quit method closes the SRP Synchronization Server. All SRP Schedule Controls are disconnected from the server before it is shutdown. Therefore, calling this method ends all SRP Schedule Control synchronization.

The Server, and Port parameters provide the information needed to find the SRP Synchronization Server. The Server is the IP or URL address describing the server's location on the network. The Port is the port upon which the SRP Synchronization Server is listening.

```
// Shutdown the server
ServerSendObj = OleCreateInstance("SRP.ScheduleSync")
If OleStatus() else
   OleCallMethod(ServerSendObj, "Quit", "127.0.0.1", "25000")
end
```