Run 8 External Dispatcher Comms Guide

1. Setting Up The Connection

To connect to Run 8 as an external dispatcher application, use the supplied DispatcherComms.dll library. This DLL is compiled with .NET 4.5. It can be found in the Run 8 installation, or in the lib folder in the sample project.

Include this DLL in your project, and add a using statement for the following namespaces:

```
// Import for Dispatcher Comms classes
using DispatcherComms;
using DispatcherComms.Run8Proxy;
using DispatcherComms.MessagesFromDispatcher;
using DispatcherComms.MessagesFromRun8;
```

The connection to Run 8 is represented by the interface DispatcherComms.IRun8. To get a concrete object of this type, use the following code:

```
// This is the correct way to get an instance of the Run 8 connection object
IRun8 mRun8 = Run8ProxyFactory.GetRun8Proxy();
```

The next step is to register for all the events that represent messages coming from Run 8:

```
// The connected and disconnected events are for information only, to let you know whether
// the connection has been established or lost. If the connection is lost, no action is
// required, it will automatically attempt to reconnect.
mRun8.Connected += HandleRun8_Connected;
mRun8.Disconnected += HandleRun8_Disconnected;

// This is sent whenever the connection is established with the latest permission
// state, or any time the state changes within Run 8. If you do not have dispatcher
// permission, you won't get any events from Run 8, and it will ignore any requests.
mRun8.DispatcherPermission += HandleRun8_DispatcherPermission;

// Data from run 8 about Blocks, Signals, Switches and Trains
mRun8.OccupiedBlocks += HandleRun8_OccupiedBlocks;
mRun8.ReversedSwitches += HandleRun8_ReversedSwitches;
mRun8.UnlockedSwitches += HandleRun8_UnlockedSwitches;
mRun8.InterlockErrorSwitches += HandleRun8_InterlockErrorSwitches;
mRun8.OccupiedSwitches += HandleRun8_OccupiedSwitches;
mRun8.Signals += HandleRun8_Signals;
mRun8.TrainData += HandleRun8_TrainData;

// Test messages coming from players over the network
mRun8.RadioTextEvent += HandleRun8_RadioTextEvent;

// DTMF codes whenever a player successfully hits a tower
mRun8.DTMFEvent += HandleRun8_DTMFEvent;

// Extra information about the current session
mRun8.SimulationState += HandleRun8_SimulationState;
```

Finally, connect to Run8 using the IP address and port:

```
// The IP address of the Run 8 instance to connect to. This should be configurable
// in the UI somewhere. To change this value after the initial call to IRun8.Start,
// you should call IRun8.SetRun8IPAddress(string), and the connection will automatically
// be closed and reopened using the new IP address
string run8IPAddress = "localhost";

// The port to use when connecting to Run 8. You should only get the port this way, as it could change
with
// a new Run 8 update.
int run8Port = DispatcherProxyFactory.DefaultExternalDispatcherPort;

// Start the attempting to connect to Run 8. This should only be called once during
// the lifetime of the application. If the connection is lost, it will automatically
// try to reconnect on its own.
mRun8.Start(run8IPAddress, run8Port);
```

2. Notes About Route IDs

Almost all messages to/from the game include a route ID to indicate which route is being referred to. Example route IDs are:

Route ID	Route Name
100	BNSF Mojave Sub
110	BNSF Needles Sub
120	BNSF Cajon Sub
130	BNSF Seligman Sub
140	CSX A Line
150	Barstow/Yermo

The list will expand as new routes are added to the game. If in doubt, log the data coming from the game to determine the correct route ID for the route you are building.

Also, with Run 8 V2 all routes that comprise a “region” are loaded together automatically. The game will cycle through the routes one by one and send status for the signals/switches/blocks/trains in each route separately.

3. Calls To Run 8

All the methods to send messages to run 8 return a bool to indicate whether or not the message made it though to the game. In testing on a LAN, messages are 100% reliable as they are sent using TCP. Generally the methods will only return false when the connection to Run 8 has been lost. It is safe to ignore this return value in practice if you are also responding to the Connected/Disconnected events.

4. The IRun8 interface

```
public interface IRun8
{
    //-----------------------------------------------------------------------------------------------
    // Messages from Run 8
    //-----------------------------------------------------------------------------------------------
    /// <summary>
    /// Sent when a connection to Run 8 is established
    /// </summary>
    event EventHandler Connected;
    /// <summary>
    /// Sent when a connection to Run 8 is lost
    /// </summary>
    event EventHandler Disconnected;

    /// <summary>
    /// Sent when the connected run 8 client is given permission to dispatch (or permission
    /// was taken away)
    /// </summary>
    event EventHandler<DispatcherPermissionEventArgs> DispatcherPermission;

    /// <summary>
    /// Updates the list of occupied blocks. Only blocks that are currently occupied will be
    /// in the list
    /// </summary>
    event EventHandler<OccupiedBlocksEventArgs> OccupiedBlocks;

    /// <summary>
    /// Updates the list of reversed switches. Only switches that are currently reversed will
    /// be in the list.
    /// </summary>
    event EventHandler<SwitchEventArgs> ReversedSwitches;

    /// <summary>
    /// Updates the list of unlocked switches. Only switches that are currently unlocked will
    /// be in the list
    /// </summary>
    event EventHandler<SwitchEventArgs> UnlockedSwitches;

    /// <summary>
    /// Updates the list of switches with an interlock error. Only switches that are currently
    /// errored will be in the list
    /// </summary>
    event EventHandler<SwitchEventArgs> InterlockErrorSwitches;

    /// <summary>
    /// Updates the list of switches that are occupied. Only switches that are currently occupied
    /// will be in the list.
    /// </summary>
    event EventHandler<SwitchEventArgs> OccupiedSwitches;

    /// <summary>
    /// Updates the list of signals. Every signal is included in every update and are
    /// referenced by index within the list.
    /// </summary>
    event EventHandler<SignalEventArgs> Signals;

    /// <summary>
    /// Updates the list of trains in the world
    /// </summary>
    event EventHandler<TrainDataEventArgs> TrainData;

    /// <summary>
    /// An incoming radio text message
    /// </summary>
    event EventHandler<RadioTextEventArgs> RadioTextEvent;

    /// <summary>
```

```
///   An incoming tower tone message
/// </summary>
event EventHandler<DTMFEventArgs> DTMFEvent;

/// <summary>
/// Updates the current simulation time and other values
/// </summary>
event EventHandler<SimulationStateEventArgs> SimulationState;

//-------------------------------------------------------------------------------------------
// Messages to Run 8
//-------------------------------------------------------------------------------------------
/// <summary>
/// Starts a thread that attempts to connect to Run 8. ONLY CALL THIS ONCE PER SESSION.
/// If the connection is lost, it will automatically attempt to reconnect.
/// </summary>
void Start(string pInitialIPAddress, int pInitialPort);

/// <summary>
/// Sets the IP address to use when looking for Run 8. This is only used to change
/// the IP after Start() has already been called
/// </summary>
void SetRun8IPAddress(string pIPAddress);

/// <summary>
/// Sets the port to use when looking for Run 8. This is only used to change the
/// port after Start() has already been called
/// </summary>
void SetRun8Port(int pPort);

/// <summary>
/// Tell Run8 to change a switch. This includes locking and unlocking a switch
/// </summary>
bool ThrowSwitch(int pRoute, int pSwitchID, ESwitchState pSwitchState);

/// <summary>
/// Tell Run8 to change a signal indication
/// </summary>
bool ChangeSignal(int pRoute, int pSignalID, ESignalIndication pSignalIndication);

/// <summary>
///   Send a text radio message from the dispatcher application
/// </summary>
bool SendRadioText(int pChannel, string pText);

/// <summary>
///   Transports the player to the given switch ID
/// </summary>
bool TransportPlayer(int pRoute, int pSwitchID);

/// <summary>
///   Transports the player to the given block detector
/// </summary>
bool TransportPlayerToBlock(int pRoute, int pBlockDetectorID);

/// <summary>
/// Forces an AI train to instantly stop. Use for emergencies!
/// </summary>
bool StopAITrain(int pTrainID);

/// <summary>
/// Tells an AI train to come to a normal Stop and hold the position. The train will
/// not move until the hold status is removed.
/// </summary>
bool HoldAITrain(int pTrainID, bool pHold);

/// <summary>
/// Tells the host to put an AI crew on a free agent train. If a player takes control of the
/// train before the host gets the message, it will be ignored.
/// </summary>
bool AIRecrewTrain(int pTrainID);
```

```
    /// <summary>
    /// Tells an AI train whether to relinquish the next time it comes to a complete stop.
    /// </summary>
    bool RelinquishAITrain(int pTrainID, bool pRelinquishWhenStopped);
}
```

5. Messages in detail

Connection Messages	Notes
`void Start(string pInitialIPAddress, int pInitialPort)`	Starts the process of connecting to Run 8. If the connection is lost, it will automatically attempt to reconnect.
`void SetRun8IPAddress(string pIPAddress)`	Used to change the IP Address after the initial call to Start
`void SetRun8Port(int pPort)`	Used to change the port after the initial call to Start

Connection Events	Notes
`event EventHandler Connected`	Fired whenever the connection to Run 8 is established
`event EventHandler Disconnected`	Fired whenever the connection to Run 8 is lost

Dispatch Messages To Run 8	Notes
`bool ThrowSwitch(int pRoute, int pSwitchID, ESwitchState pSwitchState)`	The enumeration ESwitchState has the following values: **Normal**-The switch is aligned for the straight leg **Reversed**-The switch is aligned for the curved leg **Locked**-The switch is locked for DS use only **Unlocked**-The switch is unlocked and can be thrown by hand in the 3D world **InterlockError**-This is only used by Run 8 to report status **NoError**- This is only used by Run 8 to report status If a switch state cannot be change (for example, lining a switch that is occupied), then Run 8 will ignore this message.
`bool ChangeSignal(int pRoute, int pSignalID, ESignalIndication pSignalIndication)`	The enumeration ESignalIndication has the following values: **Stop**-The signal shows all red aspect **Proceed**-The signal shows a non red aspect based on block occupancy, and will change to Stop when the signal heads in the 3D world are all red **Fleet**-The same as proceed except the signal will stay clear after a train passes **FlagBy**-The signal in the 3D world will show a restricting aspect
`bool SendRadioText(int pChannel, string pText)`	Sends a text chat message
`bool TransportPlayer(int pRoute, int pSwitchID)`	Transports the Avatar of the current player in the 3D world to the given switch
`bool TransportPlayerToBlock(int pRoute, int pBlockDetectorID)`	Transports the Avatar of the current player in the 3D world to the given block section

`bool StopAITrain(int pTrainID)`	Tells an AI crewed train to instantly stop. Once stopped, it will “hold” for dispatcher.
`bool HoldAITrain(int pTrainID, bool pHold)`	If the pHold parameter is set to true, this tells and AI crewed train to come to a normal stop then “hold” for dispatcher. If it is set to false, it will release an AI train to resume normal behaviour.
`bool AIRecrewTrain(int pTrainID)`	Instructs Run 8 to add an AI crew to an unoccupied train
`bool RelinquishAITrain(int pTrainID, bool pRelinquishWhenStopped)`	Instructs an AI crew to relinquish the train the next time it is stopped. Note that this will not cause the train to actually stop. To do that, use a Stop signal, or set the Hold flag.

Status Events From Run 8	**Notes**

`event EventHandler<DispatcherPermissionEventArgs> DispatcherPermission`	Fired to indicate the permission that the player has. It contains 2 values:
	
	`EDispatcherPermission Permission - Indicates whether dispatch permission has been granted to the player. If the player does not have dispatch permission, no data will come from Run 8, and requests to the game will be ignored`
	
	`bool AIPermission - Indicates whether the player has AI permission. Without AI permission, StopAITrain, HoldAITrain, AIRecrewTrain and RelinquishAITrain requests will be ignored`

`event EventHandler<OccupiedBlocksEventArgs> OccupiedBlocks`	Contains a list of all the block detector IDs for a single route that are occupied. Any block detector IDs for that route that are not in the list are assumed to be unoccupied.
`event EventHandler<SwitchEventArgs> ReversedSwitches`	Sends a list of all the dispatcher controlled switches that are currently reversed for a single route. Any switches in that route that are not in the list are assumed to be lined normally
`event EventHandler<SwitchEventArgs> UnlockedSwitches`	Contains a list of all the dispatcher controlled switches that are currently unlocked for a single route. Any switches in that route that are not in the list are assumed to be locked
`event EventHandler<SwitchEventArgs> InterlockErrorSwitches`	Contains a list of all the dispatcher controlled switches that are currently in an interlock error state for a single route. Any switches in that route that are not in the list are assumed to be lined correctly

`event EventHandler<SwitchEventArgs> OccupiedSwitches`	Contains a list of all the dispatcher controlled switches that are currently occupied for a single route. Any switches in that route that are not in the list are assumed to be unoccupied. This is a new feature in Run 8 V2 where the dispatch screen shows switch icons in a different colour if they are occupied. Also, the switch within Run 8 will continue in this state for several seconds after the track sections in the switch are unoccupied.
`event EventHandler<SignalEventArgs> Signals`	Contains a list of all the signals in a single route and their current state. Each time this message is sent, it will contain an entry in the array for each dispatcher controlled signal for that route. Valid states for each signal are in the enumeration ESignalIndication, described in the ChangeSignal method above.

`event EventHandler<TrainDataEventArgs> TrainData`	Contains details about a train in Run 8. The data that is sent in this message is:
	
	`int AxleCount`
	
	`int BlockID - The block detector index occupied by the lead loco of the train's control unit. This will be -1 if the train is not on a track section attached to a block detector`
	
	`string EngineerName - the player name in control of the train, or null for an unoccupied or AI engineer`
	
	`EEngineerType EngineerType - An enumeration indicating the type of crew in control of the train: values are None, Player or AI.`
	
	`float HpPerTon`
	
	`int LocoNumber - of the lead rail unit`
	
	`string RailroadInitials - of the lead rail unit`
	
	`int TrainID - an integer identifying this train within Run 8. This should be used to identify the train from one update to the next, and also in the StopAITrain, HoldAITrain, AIRecrewTrain and RelinquishAITrain requests to Run 8.`
	
	`int TrainLengthFeet`
	
	`int TrainSpeedLimitMPH - New for Run 8 to make AI trains work, track and trains speed limits are implemented. This value for all trains is passed to the external dispatcher to embarrass players who are speeding. It is based on the lower of the track speed limit and the train speed limit. It does not factor in signal speed`

	limits. float TrainSpeedMph string TrainSymbol int TrainWeightTons bool HoldingForDispatcher - Indicates whether an AI train has the "Hold" for dispatcher flag set. If so, it will come to a stop and not move again until the flag is removed bool RelinquishWhenStopped - Indicates whether an AI crew will relinquish a train the next time it happens to come to a stop.
event EventHandler<RadioTextEventArgs> RadioTextEvent	A text chat message from a player
event EventHandler<DTMFEventArgs> DTMFEvent	Indicates a player in a train has successfully toned a radio tower.
event EventHandler<SimulationStateEventArgs> SimulationState	Contains the following data: bool IsClient - A flag to indicate whether this instance of Run 8 is a client on a network. If this is false, then it is either single player or hosting a session. The intent of this value is for when AI dispatching logic has been developed, to limit when it can be used. If multiple clients attempt to use 3rd party dispatching applications that have automated dispatching logic, then they will most likely fight it out with disastrous outcomes during a network session DateTime SimulationTime - the sim time within Run 8

6. Data Classes In Detail

```
public enum ESwitchState
{
    Normal,
    Reversed,
    Locked,
    Unlocked,
    InterlockError,
    NoError
}

public class DispatcherPermissionEventArgs : EventArgs
{
    public EDispatcherPermission Permission;
    public bool AIPermission;
}

public enum EDispatcherPermission
{
    NoChange,
    Granted,
    Rescinded,
    Observer // Not currently used
}

public class OccupiedBlocksEventArgs : EventArgs
{
    public int Route { get; set; }
    public List<int> OccupiedBlocks { get; set; }
    public List<int> OpenManualSwitchBlocks { get; set; }
}

public class SwitchEventArgs : EventArgs
{
    public int Route { get; set; }
    public List<int> SwitchIDs { get; set; }
}

public class SignalEventArgs : EventArgs
{
    public int Route { get; set; }
    public List<ESignalIndication> Signals { get; set; }
}

public enum ESignalIndication
{
    Stop,
    Proceed,
    Fleet,
    FlagBy,
}
```

```
public class TrainDataEventArgs : EventArgs
{
    public TrainData Train { get; set; }
}

public class TrainData
{
    public int TrainID { get; set; }
    public int BlockID { get; set; }
    public EEngineerType EngineerType { get; set; }
    public string EngineerName { get; set; }
    public string RailroadInitials { get; set; }
    public int LocoNumber { get; set; }
    public string TrainSymbol { get; set; }
    public int TrainLengthFeet { get; set; }
    public int AxleCount { get; set; }
    public float HpPerTon { get; set; }
    public int TrainWeightTons { get; set; }
    public float TrainSpeedMph { get; set; }
    public bool HoldingForDispatcher { get; set; }
    public bool RelinquishWhenStopped { get; set; }
    public int TrainSpeedLimitMPH { get; set; }
}

public enum EEngineerType
{
    None,
    Player,
    AI
}

public class RadioTextEventArgs : EventArgs
{
    public int Channel { get; set; }
    public string Text { get; set; }
}

public class DTMFEventArgs : EventArgs
{
    public int Channel { get; set; }
    public string Tone { get; set; }
    public string TowerDescription { get; set; }
    public EDTMFType DTMFType { get; set; }
}

public enum EDTMFType
{
    None,
    DispatchTone,
    EmergencyTone
}

public class SimulationStateEventArgs : EventArgs
{
    public DateTime SimulationTime { get; set; }
    public bool IsClient { get; set; }
}
```